1992

A dynamically adaptive mesh method for internal flows

John William Slater
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9311535

A dynamically adaptive mesh method for internal flows

Slater, John William, Ph.D.

Iowa State University, 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

A dynamically adaptive mesh method for internal flows

by

John William Slater

A Dissertation Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Aerospace Engineering and Engineering Mechanics
Major: Aerospace Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1992

DEDICATION

To Auntie, Cecelia G. Slater Pochedly

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Uppercase Latin Symbols

A	Jacobian matrix for E
A^+	van Driest damping constant
$\hat{A}$	Jacobian matrix for $\hat{E}$
B	Jacobian matrix for F
$\hat{B}$	Jacobian matrix for $\hat{F}$
$\mathbf{C}$	Convective portion of the flux dyadic
C_1	Constant for Sutherland's formula
C_2	Constant for Sutherland's formula
C_{CP}	Constant for the Baldwin-Lomax turbulence model
C_{kleb}	Constant for the Baldwin-Lomax turbulence model
C_{wk}	Constant for the Baldwin-Lomax turbulence model
C_P	Specific heat at constant pressure
$\mathbf{D}$	Non-convective portion of the flux dyadic
E	Inviscid flux vector in x-coordinate direction
E_t	Total energy per unit volume
E_V	Viscous flux vector in x-coordinate direction
F	Cartesian flux vector in y-coordinate direction

F_{kleb}	Klebanoff intermittency factor
$\hat{F}$	Flux vector in generalized coordinates
$\mathbf{F}$	Cartesian flux dyadic
F_V	Viscous flux vector in y-coordinate direction
$\mathbf{H}$	Flux dyadic
I	Identity matrix
I	Variational integral
J	Jacobian
K	Clauser constant
K	Average value of WJ
L	Length
L	Lagrangian
M	Number of cell faces of a finite-volume
P	Similarity transformation matrix of right eigenvectors
P^{-1}	Similarity transformation matrix of left eigenvectors
Pr	Prandtl number
R	Gas constant
$\hat{R}$	Sum of cell-face fluxes for a cell
Re	Reynolds number
S	Surface area
T	Static temperature
U	Vector of conservative variables
$\hat{U}$	Vector of conservative variables in generalized space
$\delta\hat{U}^n$	$\hat{U}^{n+1} - \hat{U}^n$

V	Volume
$\vec{V}$	Velocity vector
W	Weight function for volume adaption

Lowercase Latin Symbols

c	Acoustic speed
e	Specific internal energy
$\vec{g}$	Mesh speed vector
h	Specific enthalpy
h_t	Total specific enthalpy
i,j	Indices for the ξ and η coordinates
$\hat{\imath}$	Unit vector in x-coordinate direction
$\hat{\jmath}$	Unit vector in y-coordinate direction
$\hat{k}$	Unit vector in z-coordinate direction
k	Thermal conductivity
k	Index
$\hat{\vec{l}}$	Left eigenvector in generalized coordinates
$\hat{n}$	Unit normal vector
p	Static pressure
$\vec{r}$	Position vector
$\hat{\vec{r}}$	Right eigenvector in generalized coordinates
s	Arclength coordinate
t	Time
u	Scalar solution variable

u	Component of velocity in x-coordinate direction
v	Component of velocity in y-coordinate direction
x	Cartesian coordinate along $\hat{\imath}$-axis
x_τ	Mesh speed in x-coordinate direction
y	Cartesian coordinate along $\hat{\jmath}$-axis
y_τ	Mesh speed in y-coordinate direction
$\vec{z}$	The mesh speed vector from the mesh speed equations

Greek Symbols

γ	Ratio of specific heats
δ	Differential
Δ	Increment
η	Transformed coordinate along $\hat{\jmath}$
ϵ	Small number
ϵ	Parameter for entropy fix
θ	Factor for explicitness and implicitness
κ	Curvature
κ	von Karman constant
λ	Eigenvalue in Cartesian coordinates
λ	Second coefficient of viscosity
$\lambda^{+/-}$	Positive/negative eigenvalues
$\hat{\lambda}$	Eigenvalue in generalized coordinates
Λ	Diagonal matrix of eigenvalues
μ	Coefficient of viscosity

ν	Courant number
ξ	Transformed coordinate along $\hat{\imath}$
ρ	Density
ω	Vorticity
ω	Relaxation factor
τ	Generalized time
ψ	Limiter function
∇	Divergence operator

Superscripts

$+$	Value associated with the positive eigenvalue
$-$	Value associated with the negative eigenvalue
n	Time level at which solution is known
$n+1$	Time level at which solution is desired
T	Transpose

Subscripts

A	Adaption
B	Boundary
dif	Difference
$extrap$	Extrapolation
G	Mesh
GB	Mesh boundary
GS	Mesh speed

$inner$	Inner layer of turbulent boundary layer
l	Laminar
L	Solution at the left of a cell face
m	Index for cell face
max	Maximum
min	Minimum
N	Normal
O	Orthogonality
$outer$	Outer layer of turbulent boundary layer
R	Solution at the right of a cell face
ref	Reference value
S	Smoothness
t	Turbulent
T	Tangential
V	Viscous term
$wall$	Wall value
x	Differentiation with respect to the x-coordinate
y	Differentiation with respect to the y-coordinate
η	Differentiation with respect to the η-coordinate
ξ	Differentiation with respect to the ξ-coordinate
τ	Differentiation with respect to time
∞	Freestream conditions
$+$	Plus direction along coordinate
$-$	Minus direction along coordinate

ACKNOWLEDGEMENTS

I would like to thank Dr. Richard G. Hindman for his guidance over the last six and one-half years as advisor and major professor. Dr. Hindman has allowed me great freedom to pursue my studies while always being ready with ideas and encouraging words when problems arose.

This work was supported through the NASA Graduate Student Researchers Program through the NASA Lewis Research Center, contract number NGT-50441. This program not only provided substantial monetary support, but also provided the computational resources needed to conduct this project. During the course of the project, I spent nearly twelve months at the Lewis Research Center. I would like to thank Dr. Meng-Sing Liou of the NASA Lewis Research Center for his support and discussions over the course of this work.

CHAPTER 1. INTRODUCTION

This dissertation is concerned with methods for the numerical computation of the unsteady fluid dynamics of compressible, viscous air behaving as a perfect gas. The physics of this flow is assumed to be mathematically modelled by the unsteady, Navier-Stokes equations. It is assumed that the significant fluid dynamics occur in planar surfaces; the flow variables are a function of only two spatial variables. The time scales of interest in the unsteady fluid dynamics are large compared to the time scales of turbulence. However, the effects of turbulence are modelled through the representation of the time-averaged stresses. This work does consider some cases involving no viscosity for which the fluid dynamics is modelled by the unsteady, Euler equations.

In a numerical approach, the continuous nature of the Navier-Stokes or Euler equations is represented on a discrete domain. This representation involves a temporal discretization and a spatial discretization.

For the temporal discretization of the unsteady, Navier-Stokes and Euler equations, the mathematical character of the equations leads to the proper approach. The Navier-Stokes equations are a mixed system of hyperbolic and parabolic equations. The unsteady, Euler equations are a system of hyperbolic equations. Both sets can be properly discretized in time by starting with an initial solution and marching in

time with a discrete time step.

For the spatial discretization, a cell-vertex, finite-volume discretization is used in which discrete mesh points are defined in the domain and the solution is defined at each mesh point. The spatial relationship between the mesh or solution points defines the structure of the mesh connecting the mesh points and thus the structure of the finite-volumes. Structured meshes define the relationship between the mesh points in a manner analogous to elements in an array. This is ideal for a computational approach. Unstructured meshes reduce the structure to involve only immediate neighbors of each mesh point. Commonly, the mesh points in an unstructured mesh form the vertices of triangles or quadrilaterals with its neighbors. This allows greater flexibility in placing the mesh points, which is useful for domains with complex geometry. The present work uses only structured meshes. This choice was based on the fact that structured meshes are more computationally efficient than unstructured meshes and that the geometries examined in this work are not extremely complex. About each mesh or solution point, a non-overlapping finite-volume is defined. The spatial character of the Navier-Stokes equations is then approximated on each finite-volume of the domain.

One should be able to imagine that the accuracy of the numerical solution is dependent on the placement of the mesh points. This leads to the notion that there is an optimum mesh for a given number of mesh points for which the numerical flow solution contains the minimum error. One can speak of the quality of the mesh; however, it is often very difficult to define an explicit expression for the solution error in terms of the characteristics of the mesh. The standard approach is to use measures of mesh quality which, when improved, will indirectly reduce the solution

error. Such measures of the mesh quality include the variation in the mesh spacing, the orthogonality of the mesh lines, and the resolution of flow solution features. This last measure indicates that the mesh quality is also dependent on the flow solution. However, one does not know the flow solution at the start of the computations. This has lead to the development of solution adaptive mesh methods. Since we are interested in structured meshes with a fixed number of mesh points, the only way the mesh can be adapted is if the mesh points are moved in relation to each other. Since the mesh points move during the time-marching process the procedure becomes a dynamically adaptive mesh method.

The approach of the present work is to use variational principles to define a measure of the mesh quality and then formulate the mesh equations from the Euler-Lagrange equations. The measure of mesh quality is a linear combination of measures of mesh smoothness, orthogonality, and volume adaption. The solution of the mesh equations then represents the optimum mesh which maximizes the mesh quality. The adaption of the mesh to the flow solution is driven by the magnitude of the gradients in the flow solution.

One can see that if a flow is unsteady, then the optimum mesh is also a function of time. Also the flow features may be rapidly changing. The solution adaptive mesh method must respond to such change. It has been common in the literature to refer to any adaptive mesh method which changes the mesh during the time-marching as a dynamically adaptive mesh method [43]. That definition is not followed here. Dynamically adaptive mesh methods are defined here to be methods which adapt the mesh as part of a time-accurate flow solution procedure. This is important for the proper computation of unsteady flows.

The focus of the present work has been to develop a dynamically adaptive mesh method coupled with a flowfield solver for the computation of unsteady, inviscid and viscous internal flows. The dynamics of the mesh is accounted for in the flow equation through the mesh speeds. Thus the flow and mesh equations are coupled and no interpolation of the solution onto the dynamic mesh is required. One approach for computing the mesh speeds is to use a backwards time difference of the mesh. Another approach is to form a system of mesh speed equations by performing the time differentiation of the mesh equations. Both approaches are investigated in the present work. The latter approach has the potential of greater accuracy because it can include the current flow dynamics in the computation of the mesh speeds.

The next chapter presents a review of dynamically adaptive mesh methods. Chapter 3 then presents the unsteady Navier-Stokes equations in the integral form. Chapter 4 begins with the integral form of the equations and discusses the finite-volume approximations. Chapter 5 discusses the mesh and mesh speed equations obtained from variational principles. Chapter 6 discusses the numerical solution of the mesh and mesh speed equations. Chapter 7 discusses the numerical methods for representing the finite-volume form of the flow equation in time and space. Chapter 8 discusses the procedure for the coupling of the flow and mesh equations. Chapter 9 presents some results for some model problems which demonstrate the behavior of the mesh and mesh speed equations. Chapter 10 presents some results for the dynamic solution adaption of the inviscid flows in a converging-diverging nozzle, viscous boundary-layer flows over a flat plate, and transonic, viscous flows in a diffuser. Chapter 11 then presents the conclusions.

CHAPTER 2. A BACKGROUND ON DYNAMICALLY ADAPTIVE MESH METHODS

This chapter reviews dynamically adaptive mesh methods for which the mesh moves during the time integration to adapt to moving boundaries or transient features in the flow. The main emphasis is on methods which adapt to unsteady flow features. A short discussion is presented on methods which use finite-element and unstructured meshes; however, the main emphasis is on structured meshes involving the movement of a fixed number of mesh points. Several survey papers have been written which include discussions on dynamically adaptive mesh methods [2] [3] [19] [23] [43].

Flow and Mesh Coupling Approaches

The primary distinction among dynamic mesh methods is the manner in which the mesh dynamics is coupled to the flow dynamics.

The most accurate level of coupling is the *completely coupled approach* in which the flow dynamic equations and the mesh dynamic equations form one set of equations which are solved simultaneously for the flow and mesh states. This approach has been applied to moving finite element methods [6]. In this approach, the interpolation function used for the representation of the flow is also used for the mesh. The combined system of equations is then solved using the method of weighted residuals.

Dorfi and Drury [18] use a completely coupled approach to implicitly solve a coupled, finite-difference system of equations for the one-dimensional shock tube problem.

The next level of coupling is the *strongly coupled approach* in which the flow and mesh dynamic equations are solved separately, but are advanced in time in such a manner that the flow state and mesh state evolve simultaneously. This requires a communication procedure between the flow equations and the mesh equations which usually involves a multi-stage or iterative method for advancing the flow and mesh states in time. This is the approach followed in the present work.

The next level of coupling is the *weakly coupled approach* in which the mesh equations are solved using flow solutions from previous time steps. Thus, there is a lag between the flow dynamics and the mesh dynamics. This is the most common adaptive mesh approach and has produced some good results for steady flows.

The lowest level of coupling is the *uncoupled approach* in which the mesh remains fixed through the entire time integration of the flow equations.

Some Requirements for a Dynamic Mesh Method for Unsteady Flows

The focus of the present work is on dynamic mesh methods in which the mesh adapts to unsteady flow features in the solution. Thus, the procedure is required to be time-accurate. This requires at least a strongly coupled approach. The mesh must accurately track the moving flow features to provide the improved resolution; however there must be proper resolution of other areas in the flow domain. If the flow feature has diminished, the clustering of the mesh should also be diminished. The dynamic mesh method should allow proper communication among the mesh points so that mesh lines do not cross and that mesh motion can be anticipated. This

suggests an elliptic nature for the mesh equations. The dynamic mesh method must be extendable to three-dimensional space.

Approaches for Communicating the Flow and Mesh Dynamics

This section discusses the details on how the flow and mesh states communicate within the flow and mesh dynamic equations.

The first approach accounts for the flow and mesh dynamics within the dynamic equations. For the flow dynamic equations, this involves assuming that the control volume may be changing shape in time, and this requires accounting for the convection due to control volume boundary motion through the boundary flux terms. When the discretization of the flow equations is carried out, this results in mesh speed terms in the flow equations. For the mesh dynamic equations, this involves accounting for the flow solution dynamics. The mesh equations include a measure of the flow solution. The mesh dynamic equations will then include a measure of the flow dynamics. This approach is followed in the present work. A review of dynamically adaptive mesh methods using this approach is presented in the next section.

The next approach for communicating the flow and mesh dynamics involves no direct accounting for the mesh dynamics. The usual approach is to advance the flow solution on a static mesh, compute a new mesh, and then perform a rezoning or interpolation of the new solution on the old mesh onto the new mesh. This approach has been used mainly for solution mesh adaption for steady flows. Any errors in the interpolation are removed as the mesh and flow solutions become steady. A weakly coupled approach can be used since time accuracy is not of importance. If time accuracy is important, then this approach requires a conservative interpolation

procedure of accuracy of the same order as the time-integration method. This approach is followed in Bockelie, Eiseman, and Smith [10] for the dynamic adaption of the mesh for one- and two-dimensional unsteady inviscid flows. A mesh predictor-corrector method temporally links the flow solver and the adaptive mesh procedure. The predictor and corrector are treated as initial value problems in which the mesh is recomputed and conservative rezonings are performed to transfer data to the new meshes. Significant in their approach is that it allows the use of a general flow solver and a general adaptive mesh procedure. The dynamically adaptive mesh procedure is demonstrated for a shocktube problem and a shock wave interacting with a vortex.

Another approach for communicating the flow and mesh dynamics is to simply neglect the mesh dynamics. As the flow solution is marched in time, a new mesh solution is periodically determined. However, no mesh speeds are computed and no interpolation is performed. Connett et al. [15] use this approach for unsteady computations. They rely on iterations at each time step to match the mesh and flow solutions. For steady flows, the iterations may not be needed. The errors associated with neglecting the mesh dynamics may be removed as the steady state is reached.

The Computation of the Mesh Speeds

Mesh speeds can be obtained from a time difference of the mesh. This requires storing one or more levels of mesh coordinates, depending on the desired accuracy of the differencing. At the initial time level when no previous mesh data exist, the mesh speeds are usually set to zero. When the differencing is part of a multi-stage or iterative integration method, it may be possible to compute the mesh speeds with the same order as the time integration method. Benson and McRae appear to use

this approach [9].

Another approach for computing the mesh speeds is to compute the mesh speeds directly and then integrate the mesh speeds to determine the mesh. These methods are known as mesh speed methods. Survey papers by Anderson [2] [3] discuss some mesh speed methods of the early 1980s.

Rai [38] and Anderson and Rai [4] discuss a mesh-speed method in which the mesh speeds are computed based on an attraction-repulsion approach. The solution error is approximated at each mesh point and the mesh speeds are computed to either attract or repel other mesh points in order to equalize the error at each mesh point. The method was applied to one- and two-dimensional meshes for fluid dynamics computations for steady flows. Anderson [3] presents a discussion of the application of the method to unsteady problems and concludes that the attraction-repulsion model does not provide an acceptable mesh adaption method because the method exhibited a memory.

Greenberg [20] proposes a mesh speed equation based on a chemical reaction analogy. Harten and Hyman [22] propose a mesh speed equation based on the wave propagation information in inviscid flows.

Hindman et al. [28] introduce the concept of obtaining a mesh speed equations from the time differentiation of the mesh equations. The method is applied for the solution of the two-dimensional, unsteady Euler equations for regions in which a shock-fitted boundary is in motion. The mesh is generated using an elliptic equation. The prescribed speeds of the boundary mesh points are used as boundary conditions for the mesh speed equations and its solution yields the mesh speeds for the interior points. There is no solution adaption. MacCormack's method is used to integrate

the flow equations and mesh speeds.

Hindman and Spencer [29] expand on the concept of Hindman et al. [28] to include solution adaption on one-dimensional meshes. A Poisson mesh equation is formulated in which the forcing function is an adaption function. The steady mesh equation is differentiated with respect to time to get a linear mesh speed equation which is solved numerically to obtain the mesh speeds. The flow equations are included in the mesh speed equation and thus provide the coupling between the solution and mesh movement. The MacCormack method is used to integrate the linear wave equation and the inviscid Burgers' equation. The mesh speeds are integrated in an analogous fashion to obtain the new mesh.

Holcomb and Hindman [32] continue the work of Hindman et al. [28] and Hindman and Spencer [29] to include solution adaption on two-dimensional meshes. A mesh speed equation is developed based on the time-differentiation of the steady Poisson mesh equation. They use the MacCormack method to solve the two-dimensional Burgers' equation on a square domain. A mesh control law is introduced to stabilize the computation of the mesh speeds.

Beck [8] extends the approach of Hindman and Spencer [29] to the one-dimensional, unsteady Euler equations for the solution of shock tube flows.

The objective of the present work is to develop a two-dimensional, dynamically adaptive mesh method based on the approach of Hindman et al. [28] and Holcomb and Hindman [32] in which mesh speed equations are derived from the time differentiation of the mesh equations. The approach is then applied to dynamically adapt meshes for solutions of the unsteady, Navier-Stokes equations.

CHAPTER 3. THE UNSTEADY NAVIER-STOKES EQUATIONS

The system of Navier-Stokes equations models the fluid dynamics of air and include the principles of the conservation of mass, momentum, and energy. These principles are applied to the material or Lagrangian representation of the fluid. However, fluid dynamics is most conveniently represented through a spatial or Eulerian representation. The Reynolds' transport theorem transforms the Lagrangian representation to the Eulerian representation. The Eulerian representation must account for the time-variation of the shape of the control volume since, for dynamically adaptive meshes, the shape of the finite-volume cells may vary in time.

The flow is assumed in this work to be planar or to vary only in the $x - y$ plane and to have no fluid velocity component normal to the plane.

The system of unsteady, Navier-Stokes equations in integral form are

$$\frac{d}{dt}\int_{V(t)} U \, dV + \oint_{S(t)} \mathbf{H} \cdot \hat{n} dS = 0. \tag{3.1}$$

The U is the algebraic vector of conservative variables,

$$U^T = (\rho, \ \rho u, \ \rho v, \ E_t) \tag{3.2}$$

where

$$E_t = \frac{p}{\gamma - 1} - \frac{1}{2}\rho \left(u^2 + v^2 \right)$$

and p, ρ, u, and v are the primitive variables of pressure, density, and the Cartesian velocity components, respectively.

The $\mathbf{H}$ is the flux dyadic, which for a time-varying control volume is

$$\mathbf{H} = \mathbf{F} - \vec{g}\, U. \tag{3.3}$$

The $\vec{g}$ is the velocity vector of the boundary of the control volume

$$\vec{g} = x_T\, \hat{i} + y_T\, \hat{j}. \tag{3.4}$$

The Cartesian flux dyadic for the two-dimensional, unsteady Navier-Stokes equations takes the form of

$$\mathbf{F} = \left(E - E_V \right) \hat{i} + \left(F - F_V \right) \hat{j}. \tag{3.5}$$

The flux dyadic can also be expressed as

$$\mathbf{H} = \mathbf{C} - \mathbf{D}, \tag{3.6}$$

where $\mathbf{C}$ is the portion of the flux dyadic containing the convective terms,

$$\mathbf{C} = \left(\vec{V} - \vec{g} \right) U, \tag{3.7}$$

and $\mathbf{D}$ is the portion of the flux dyadic containing the non-convective terms.

The algebraic vectors for the inviscid fluxes are

$$E^T = \left(\rho u,\ \ p + \rho u^2,\ \ \rho uv,\ \ \left(p + E_t \right) u \right) \tag{3.8}$$

and

$$F^T = \left(\rho v,\ \ \rho uv,\ \ p + \rho v^2,\ \ \left(p + E_t \right) v \right). \tag{3.9}$$

The algebraic vectors for the viscous fluxes are

$$E_V = \begin{pmatrix} 0 \\ \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) + 2 \mu \frac{\partial u}{\partial x} \\ \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ k \frac{\partial T}{\partial x} + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) u + 2 \mu \frac{\partial u}{\partial x} u + \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) v \end{pmatrix} \tag{3.10}$$

and

$$F_V = \begin{pmatrix} 0 \\ \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) + 2 \mu \frac{\partial v}{\partial y} \\ k \frac{\partial T}{\partial y} + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) v + 2 \mu \frac{\partial v}{\partial y} v + \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) u \end{pmatrix}. \tag{3.11}$$

The equation of state for a perfect gas is

$$p = \rho R T, \tag{3.12}$$

where R is the gas constant, which for air is $R = 287.0\ m^2 / s^2\ K$.

The unsteady, Navier-Stokes equations govern the fluid dynamics of turbulent flow. However, the space and time scales of turbulence are much finer than those encountered in the numerical approximations of this work. Turbulence is modelled through a mass-weighted averaging of the flow variables over a time interval which is long compared to the time scale of the turbulence, but which is short compared to the time scale of the numerical approximation [5]. The Boussinesq assumption is used which assumes the Reynolds stresses and heat fluxes occuring in the time-averaged Navier-Stokes equations are of the same form as the laminar stresses and heat fluxes, repectively. Thus, the μ of equations (3.10) and (3.11) is replaced by

$$\mu = \mu_l + \mu_t \tag{3.13}$$

and the thermal conductivity is replaced by

$$k = k_l + k_t. \tag{3.14}$$

Stokes' hypothesis is used to assign values for the second coefficient of viscosity,

$$\lambda = -2\,\mu / 3. \tag{3.15}$$

The laminar viscosity is computed using Sutherland's formula,

$$\mu_l = C_1\, T^{3/2} / (\, T + C_2\,), \tag{3.16}$$

where for air, the constants are $C_1 = 1.458x10^{-6}\ kg\ /\ (m\ s\ K^{1/2})$ and $C_2 = 110.4\ K$. The thermal conductivity is computed using the definition of the Prandtl number,

$$Pr = \mu\, c_p\ /\ k. \tag{3.17}$$

Thus,

$$k = c_p \left(\mu_l\ /\ Pr_l + \mu_t\ /\ Pr_t \right). \tag{3.18}$$

The Prandtl numbers are assumed to be constants, and for air, $Pr_l = 0.72$ and $Pr_t = 0.9$. The specific heat at constant pressure is computed for a perfect gas as

$$c_p = \gamma\, R\ /\ (\gamma - 1) \tag{3.19}$$

where for air, $\gamma = 1.4$.

The Baldwin-Lomax algebraic turbulence model [7] is used to compute the turbulent viscosity, μ_t. It is a two-layer model of the form,

$$\mu_t = \begin{cases} (\mu_t)_{inner} & \text{if } s \le s_{crossover} \\ (\mu_t)_{outer} & \text{if } s > s_{crossover}\ . \end{cases} \tag{3.20}$$

The s is the arclength along the coordinate line normal to the wall with $s = 0$ at the surface of the wall and s increasing outward from the wall. The $s_{crossover}$ is the value of s at which $(\mu_t)_{inner}$ first exceeds the value of $(\mu_t)_{outer}$. The Prandtl-Van Driest formulation is used in the inner layer

$$(\mu_t)_{inner} = \rho\, l^2 \mid \omega \mid, \tag{3.21}$$

where

$$l = \kappa\, s \left[1 - \exp(- s^+/A^+) \right]. \tag{3.22}$$

The $\mid \omega \mid$ is the magnitude of the vorticity,

$$\mid \omega \mid = \mid \partial v/\partial x - \partial u/\partial y \mid \tag{3.23}$$

and

$$s^+ = s \left(\rho_{wall}\, \tau_{wall} \right)^{1/2} / \mu_{wall}. \tag{3.24}$$

The von Kármán constant is set to a value of $\kappa = 0.4$ and the van Driest damping constant is set to a value of $A^+ = 26$. The subscript *wall* denotes the values evaluated at the wall. The shear stress at the wall is

$$\tau_{wall} = \mu_{wall} \left| \partial V_T/\partial s \right|_{wall} \tag{3.25}$$

where V_T is the magnitude of the velocity tangential to the wall.

The outer-layer model is

$$(\mu_t)_{outer} = K\, C_{cp}\, \rho\, F_{wake}\, F_{kleb}(s). \tag{3.26}$$

The Clauser constant is set to a value of $K = 0.0168$ and the constant $C_{cp} = 1.6$. The F_{wake} is defined by

$$F_{wake} = \min \left\{ \begin{array}{l} s_{max}\, F_{max} \\ C_{wk}\, s_{max}\, U_{dif}^2 / F_{max} \end{array} \right\} \tag{3.27}$$

where the constant $C_{wk} = 0.25$. The F_{max} is the maximum of the function

$$F(s) = s \mid \omega \mid \left[1 - \exp(- s^{+}/A^{+}) \right] \tag{3.28}$$

and s_{max} is the value of s at which it occurs. The U_{dif} is the maximum difference in the magnitudes of the velocities in the velocity profile. The $F_{kleb}(s)$ is the Klebanoff intermittency factor expressed as

$$F_{kleb}(s) = \left[1 + 5.5 \left(C_{kleb}\, s \,/\, s_{max} \right)^{6} \right]^{-1}, \tag{3.29}$$

where $C_{kleb} = 0.3$.

The Baldwin-Lomax turbulence model is applicable in attached and separated boundary layers and wakes. The values of the constants are from reference [7] and are applicable for the case of a constant pressure boundary layers at transonic speeds.

The system of Navier-Stokes equations is nondimensionalized so as to normalize the values to improve accuracy. The fundamental reference variables are temperature, density, and a reference length. The velocities are nondimensionalized by the acoustic speed computed from the reference temperature.

When it can be assumed that there is no heat conduction and no viscous dissipation, the equations simplify to the unsteady, Euler equations.

The unsteady, Navier-Stokes equations are a mix of hyperbolic and parabolic equations. The dissipation terms in the momentum and energy equations introduce damping into the wave behavior of the solution. The unsteady, Euler equations are a system of hyperbolic equations. Thus, the mathematical character allows for numerical methods which march in time using a discrete time step.

CHAPTER 4. THE FINITE-VOLUME FORMULATION

The system of unsteady, Navier-Stokes equations was presented in the previous chapter as

$$\frac{d}{dt}\int_{V(t)} U \, dV + \oint_{S(t)} \mathbf{H} \cdot \hat{n} dS = 0. \tag{4.1}$$

The present chapter discusses how the spatial integrals are approximated for the cell-vertex, finite-volume approach.

The Cell-Vertex, Finite-Volume Mesh

The structured mesh approach orders the location of the mesh points in a manner analogous to the elements in an array. The two-dimensional, structured mesh has dimensions of NI by NJ, (NIxNJ), mesh points. A mesh point is denoted by its (i,j) indices. The procedures for generating the mesh point locations are presented in Chapter 5. This section introduces some of the concepts involved in defining a cell-vertex, finite-volume mesh.

The two-dimensional, structured mesh obtained from the mesh generation procedure is referred to as the primary mesh. The mesh points are located at the vertices of this mesh. For a cell-vertex approach, the solution point is also located at the vertices of the primary mesh. A secondary mesh is defined by connecting the midpoints of the straight, mesh lines connecting the solution points. Figure 4.1 shows

an example of a primary mesh with the solution points at the vertices and the secondary mesh defined from the primary mesh. The encirlement of a solution point by the secondary mesh lines defines the finite-volume cell for that solution point. Thus, for the two-dimensional, structured mesh, the finite-volume cell is a quadrilaterial in which each side or cell face consists of two straight-line segments. This definition of the finite-volume cell will place the cell faces approximately midway between the solution points.

The Finite-Volume Approximations

In the finite-volume approach, the spatial integrals of the Navier-Stokes equations are approximated on each of the finite-volume cells. The fundamental approximation is that the solution over a finite-volume is represented by an average value for the entire finite-volume cell,

$$\int_V U \, dV \approx (U \, V) = \hat{U}. \tag{4.2}$$

The flux vector for a cell face is defined as

$$\hat{F} = \mathbf{H} \cdot \hat{n} dS. \tag{4.3}$$

where $\hat{n}dS$ is the cell-face area vector directed normal to the cell face and in the direction of the respective coordinate line. Figure 4.2 shows the flux vectors for the finite-volume about the solution point (i, j). The magnitude of $\hat{n}dS$ is the area of the cell face. The surface integral of the cell-face flux vector can be approximated as

$$\oint_{S(t)} \mathbf{H} \cdot \hat{n} dS \approx \sum_{m=1}^{M} \hat{F}(\tau)_m. \tag{4.4}$$

where the subscript m denotes a particular cell face and M is the total number of cell faces of the finite-volume cell. The present work assumes that the finite-volume cells are quadrilaterals for which $M = 4$.

For later discussions, let the sum of the cell-face fluxes be represented as

$$\hat{R} = \sum_{m=1}^{M=4} \hat{F}_m. \tag{4.5}$$

Thus, for the finite-volume cell (i, j), the sum of the fluxes is

$$\hat{R}_{i,j} = \hat{F}_{i+1/2,j} - \hat{F}_{i-1/2,j} + \hat{F}_{i,j+1/2} - \hat{F}_{i,j-1/2}. \tag{4.6}$$

The substitution of these approximations into equation 4.1 results in the semi-discrete finite-volume equation for the finite-volume cell (i, j),

$$\frac{d}{d\tau}\hat{U}(\tau)_{i,j} + \hat{R}(\tau)_{i,j} = 0. \tag{4.7}$$

The time discretization of equation 4.7 will be discussed in Chapter 7.

Equation (4.7) is an approximation of the spatial integral equation (4.1) which represents a set of conservation principles. It represents the notion that the time rate of change of U in the volume is determined by the net flux through the surface bounding the volume. This conservation principle must be represented in the numerical method to properly compute all solutions, especially solutions with discontinuities (i.e., weak solutions) [26]. It can be directly seen that the finite-volume representation, equation (4.7), does maintian this conservative property.

The Finite-Volume Cell-Face Area Vector

This section discusses how the cell-face area vector, $\hat{n}dS$, is computed from the geometry of the secondary mesh. Figure 4.2 shows the geometry of the secondary

mesh about the solution point (i,j).

The cell-face area vector is represented as the vector sum of the two straight-line segments that compose the cell face,

$$\hat{n}dS \;=\; (\hat{n}dS)_{+} \;+\; (\hat{n}dS)_{-} \tag{4.8}$$

where the subscripts + and − denote the two segments relative to a coordinate direction. Thus

$$\hat{n}dS_{i+1/2} \;=\; \hat{n}dS_{(i+1/2)+} \;+\; \hat{n}dS_{(i+1/2)-} \tag{4.9}$$

where

$$\hat{n}dS_{(i+1/2)-} \;=\; \frac{1}{2}\left(\vec{r}_{i+1/2,j} - \vec{r}_{i+1/2,j-1}\right) \times \hat{k} \tag{4.10}$$

and

$$\hat{n}dS_{(i+1/2)+} \;=\; \frac{1}{2}\left(\vec{r}_{i+1/2,j+1} - \vec{r}_{i+1/2,j}\right) \times \hat{k}. \tag{4.11}$$

The position vectors follow the form

$$\vec{r}_{i+1/2,j} \;=\; \frac{1}{2}\left(x_{i,j} + x_{i+1,j}\right)\hat{\imath} \;+\; \frac{1}{2}\left(y_{i,j} + y_{i+1,j}\right)\hat{\jmath}. \tag{4.12}$$

Thus, the cell-face area vectors become

$$\begin{aligned} \hat{n}dS_{(i+1/2)-} \;=\; \frac{1}{4}\;\Big[\;& \left(y_{i,j} + y_{i+1,j} - y_{i,j-1} - y_{i+1,j-1}\right)\hat{\imath} \\ -\;& \left(x_{i,j} + x_{i+1,j} - x_{i,j-1} - x_{i+1,j-1}\right)\hat{\jmath}\;\Big] \end{aligned} \tag{4.13}$$

and

$$\begin{aligned} \hat{n}dS_{(i+1/2)+} \;=\; \frac{1}{4}\;\Big[\;& \left(y_{i,j+1} + y_{i+1,j+1} - y_{i,j} - y_{i+1,j}\right)\hat{\imath} \\ -\;& \left(x_{i,j+1} + x_{i+1,j+1} - x_{i,j} - x_{i+1,j}\right)\hat{\jmath}\;\Big]. \end{aligned} \tag{4.14}$$

Similarly for the cell face in the η-coordinate direction,

$$\hat{n}dS_{j+1/2} = \hat{n}dS_{(j+1/2)+} + \hat{n}dS_{(j+1/2)-} \tag{4.15}$$

where

$$\hat{n}dS_{(j+1/2)-} = \frac{1}{2}\left(\vec{r}_{i,j+1/2} - \vec{r}_{i-1,j+1/2}\right) \times \hat{k} \tag{4.16}$$

and

$$\hat{n}dS_{(j+1/2)+} = \frac{1}{2}\left(\vec{r}_{i+1,j+1/2} - \vec{r}_{i,j+1/2}\right) \times \hat{k}. \tag{4.17}$$

The position vectors follow the form

$$\vec{r}_{i,j+1/2} = \frac{1}{2}\left(x_{i,j} + x_{i,j+1}\right)\hat{\imath} + \frac{1}{2}\left(y_{i,j} + y_{i,j+1}\right)\hat{\jmath}. \tag{4.18}$$

Thus, the cell-face area vectors become

$$\begin{aligned}\hat{n}dS_{(j+1/2)-} = \frac{1}{4} \; [\; & \left(y_{i-1,j} + y_{i-1,j+1} - y_{i,j} - y_{i,j+1}\right)\hat{\imath} \\ - \; & \left(x_{i-1,j} + x_{i-1,j+1} - x_{i,j} - x_{i,j+1}\right)\hat{\jmath} \;]\end{aligned} \tag{4.19}$$

and

$$\begin{aligned}\hat{n}dS_{(j+1/2)+} = \frac{1}{4} \; [\; & \left(y_{i,j} + y_{i,j+1} - y_{i+1,j} - y_{i+1,j+1}\right)\hat{\imath} \\ - \; & \left(x_{i,j} + x_{i,j+1} - x_{i+1,j} - x_{i+1,j+1}\right)\hat{\jmath} \;].\end{aligned} \tag{4.20}$$

The Finite-Volume Cell Volume

This section discusses how the cell volume, V, is computed from the secondary mesh geometry. The volume of the cell, which is actually an area for a planar cell, is computed by summing up the volumes of the eight triangles which make up the

cell. Figure 4.3 shows how the finite-volume cell is divided into the eight triangles. In general the volume of a triangle can be determined as

$$V = \frac{1}{2} \mid \vec{r}_a \times \vec{r}_b \mid \tag{4.21}$$

where $\vec{r}_a$ and $\vec{r}_b$ are any two sides of the triangle. Figure 4.3 shows how the vectors are defined. All of the vectors are directed in the positive coordinate direction.

The vectors to the midpoints are

$$\vec{r}_A = \frac{1}{2}\,\vec{r}_{(i+1)/i,j} = \frac{1}{2}\left[\,(x_{i+1,j} - x_{i,j})\,\hat{\imath} + (y_{i+1,j} - y_{i,j})\,\hat{\jmath}\,\right], \tag{4.22}$$

$$\vec{r}_B = \frac{1}{2}\,\vec{r}_{i,(j+1)/j} = \frac{1}{2}\left[\,(x_{i,j+1} - x_{i,j})\,\hat{\imath} + (y_{i,j+1} - y_{i,j})\,\hat{\jmath}\,\right], \tag{4.23}$$

$$\vec{r}_C = \frac{1}{2}\,\vec{r}_{i/(i-1),j} = \frac{1}{2}\left[\,(x_{i,j} - x_{i-1,j})\,\hat{\imath} + (y_{i,j} - y_{i-1,j})\,\hat{\jmath}\,\right], \tag{4.24}$$

and

$$\vec{r}_D = \frac{1}{2}\,\vec{r}_{i,j/(j-1)} = \frac{1}{2}\left[\,(x_{i,j} - x_{i,j-1})\,\hat{\imath} + (y_{i,j} - y_{i,j-1})\,\hat{\jmath}\,\right]. \tag{4.25}$$

The vectors $\vec{r}_k$ for $(k = 1, ..., 8)$ are the average of the vectors to the midpoints for the two bracketing mesh lines. Thus

$$\vec{r}_1 = \frac{1}{4}\left[\vec{r}_{i,(j+1)/j} + \vec{r}_{i+1,(j+1)/j}\right] \tag{4.26}$$

and

$$\vec{r}_2 = \frac{1}{4}\left[\vec{r}_{(i+1)/i,j+1} + \vec{r}_{(i+1)/i,j}\right]. \tag{4.27}$$

The other vectors follow the same logic.

The volumes can then be computed and the results are as follows:

$$\begin{aligned} V_1 = \frac{1}{16} \;\mid\; & (x_{i+1,j} - x_{i,j})(y_{i,j+1} - y_{i,j} + y_{i+1,j+1} - y_{i+1,j}) \\ - \; & (y_{i+1,j} - y_{i,j})(x_{i,j+1} - x_{i,j} + x_{i+1,j+1} - x_{i+1,j}) \mid, \end{aligned} \tag{4.28}$$

$$V_2 = \frac{1}{16} \; | \; (x_{i,j+1} - x_{i,j})(y_{i+1,j+1} - y_{i,j+1} + y_{i+1,j} - y_{i,j}) - (y_{i,j+1} - y_{i,j})(x_{i+1,j+1} - x_{i,j+1} + x_{i+1,j} - x_{i,j}) \; |, \quad (4.29)$$

$$V_3 = \frac{1}{16} \; | \; (x_{i,j+1} - x_{i,j})(y_{i,j+1} - y_{i-1,j+1} + y_{i,j} - y_{i-1,j}) - (y_{i,j+1} - y_{i,j})(x_{i,j+1} - x_{i-1,j+1} + x_{i,j} - x_{i-1,j}) \; |, \quad (4.30)$$

$$V_4 = \frac{1}{16} \; | \; (x_{i,j} - x_{i-1,j})(y_{i-1,j+1} - y_{i-1,j} + y_{i,j+1} - y_{i,j}) - (y_{i,j} - y_{i-1,j})(x_{i-1,j+1} - x_{i-1,j} + x_{i,j+1} - x_{i,j}) \; |, \quad (4.31)$$

$$V_5 = \frac{1}{16} \; | \; (x_{i,j} - x_{i-1,j})(y_{i-1,j} - y_{i-1,j-1} + y_{i,j} - y_{i,j-1}) - (y_{i,j} - y_{i-1,j})(x_{i-1,j} - x_{i-1,j-1} + x_{i,j} - x_{i,j-1}) \; |, \quad (4.32)$$

$$V_6 = \frac{1}{16} \; | \; (x_{i,j} - x_{i,j-1})(y_{i,j} - y_{i-1,j} + y_{i,j-1} - y_{i-1,j-1}) - (y_{i,j} - y_{i,j-1})(x_{i,j} - x_{i-1,j} + x_{i,j-1} - x_{i-1,j-1}) \; |, \quad (4.33)$$

$$V_7 = \frac{1}{16} \; | \; (x_{i,j} - x_{i,j-1})(y_{i+1,j} - y_{i,j} + y_{i+1,j-1} - y_{i,j-1}) - (y_{i,j} - y_{i,j-1})(x_{i+1,j} - x_{i,j} + x_{i+1,j-1} - x_{i,j-1}) \; |, \quad (4.34)$$

and

$$V_8 = \frac{1}{16} \; | \; (x_{i+1,j} - x_{i,j})(y_{i,j} - y_{i,j-1} + y_{i+1,j} - y_{i+1,j-1}) - (y_{i+1,j} - y_{i,j})(x_{i,j} - x_{i,j-1} + x_{i+1,j} - x_{i+1,j-1}) \; | \, . \quad (4.35)$$

The volume of the finite-volume cell (i, j) is

$$V_{i,j} \; = \; \sum_{k=1}^{8} V_k. \quad (4.36)$$

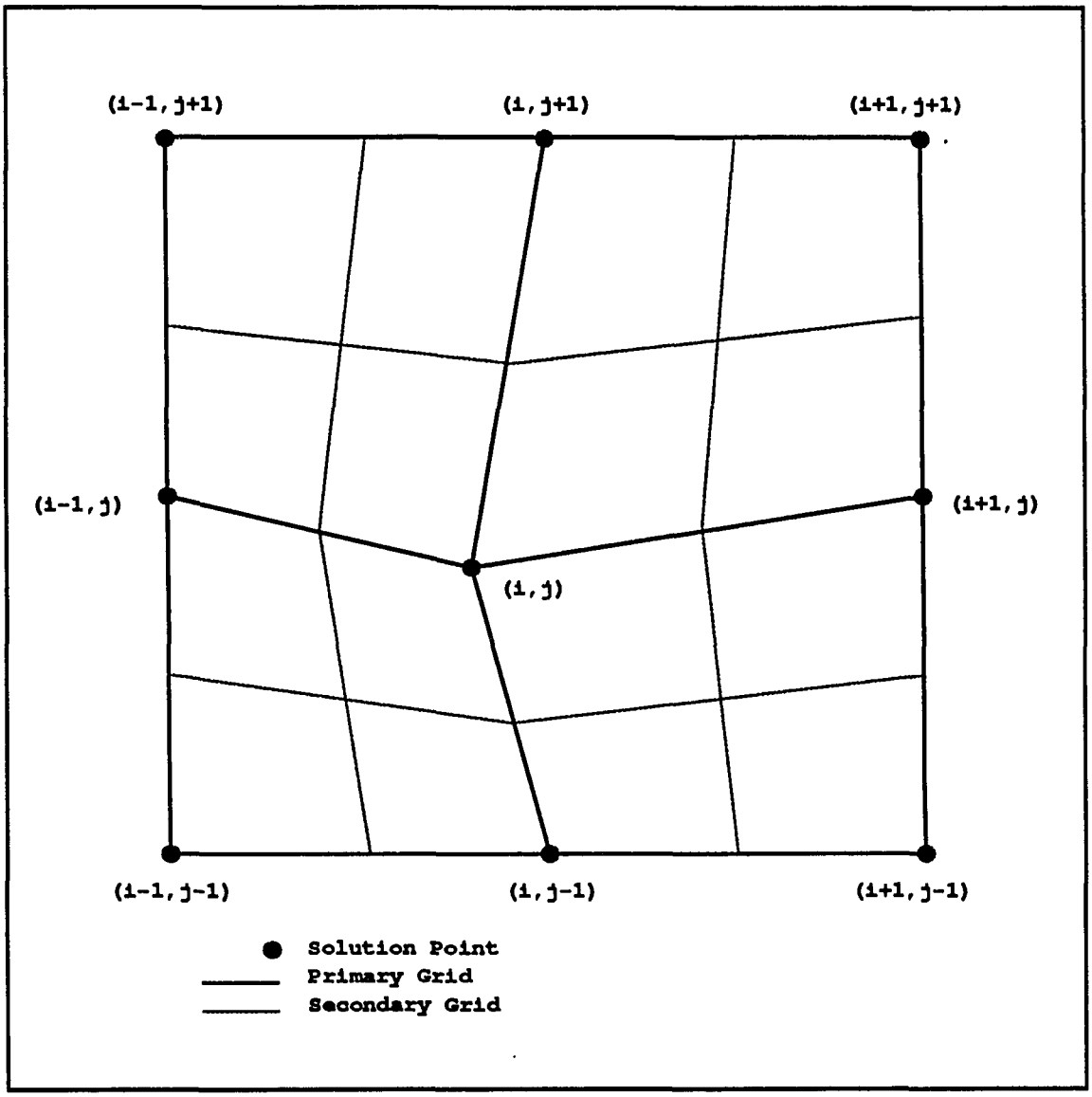

Figure 4.1: The primary and secondary meshes for a two-dimensional cell-vertex, finite-volume mesh

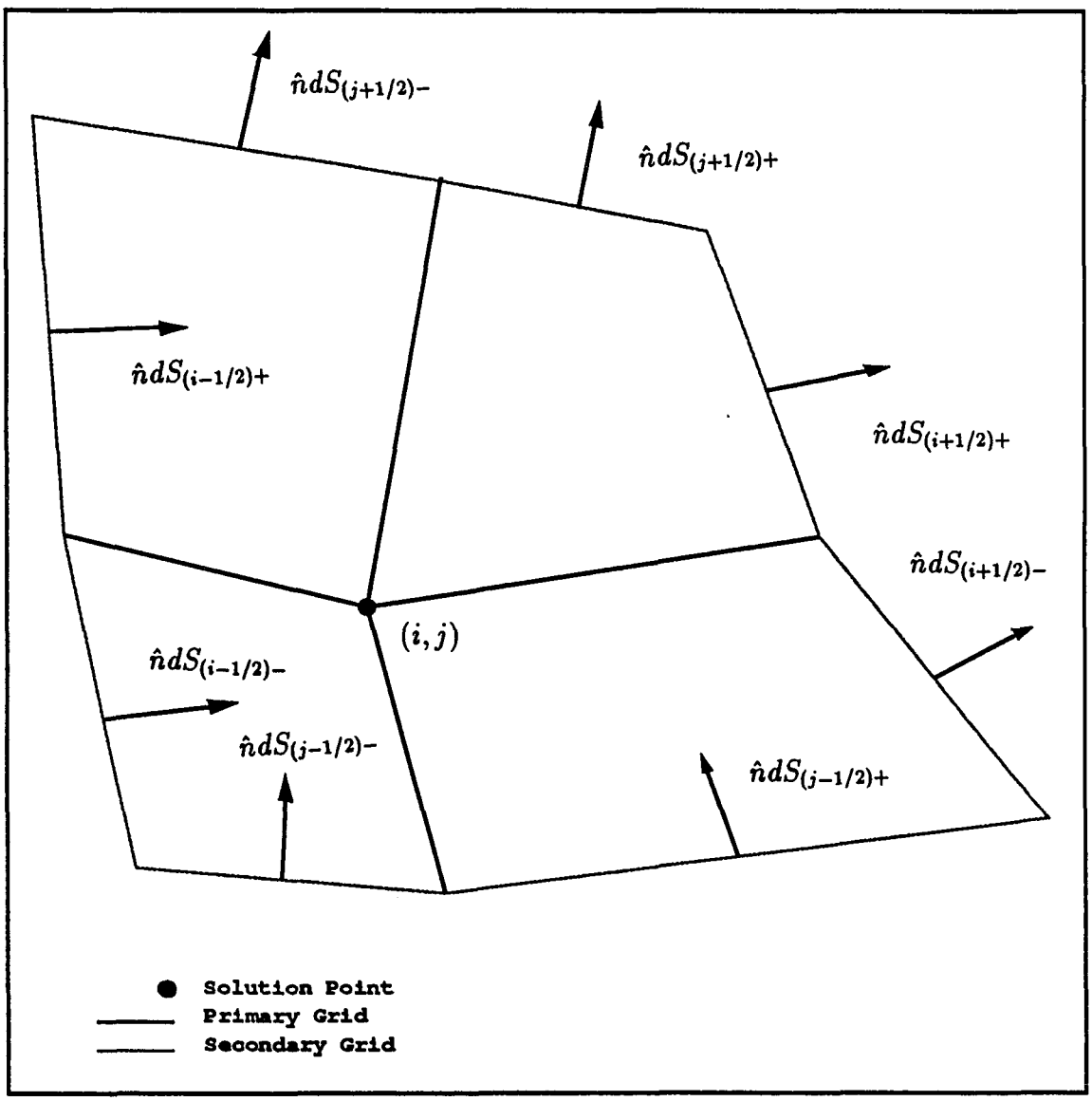

Figure 4.2: The geometry of the secondary mesh about the solution point (i, j)

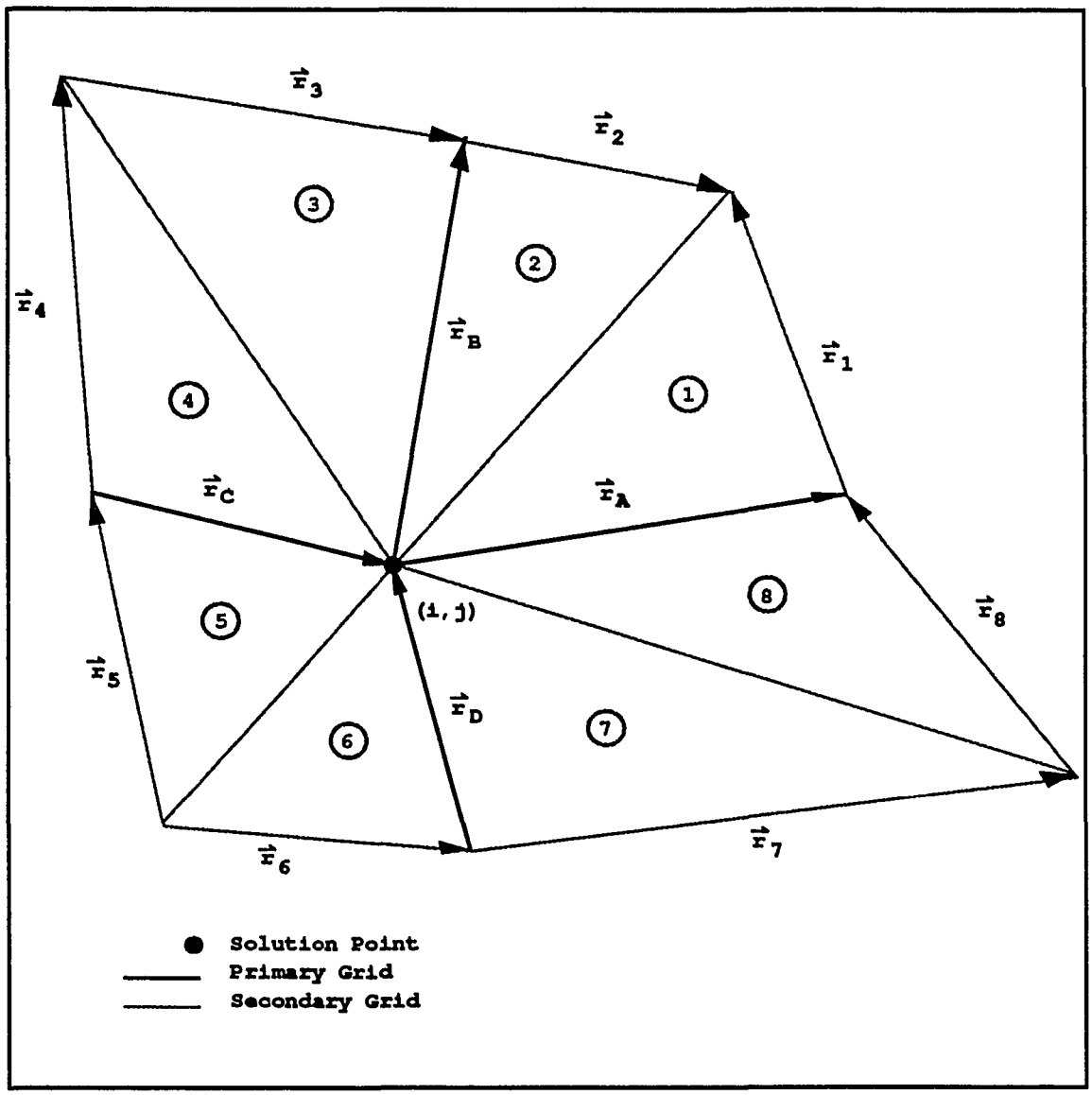

Figure 4.3: The two-dimensional finite-volume cell for cell (i, j) showing the cell volume sub-volumes

CHAPTER 5. THE MESH AND MESH SPEED EQUATIONS

This chapter discusses the development of the mesh and mesh speed equations using a variational approach. The mesh equations compute the locations of the mesh points. There have been numerous survey papers, [1], [19], [23], [42], [43]; a conference proceeding [14]; and a textbook [44] written and compiled on various aspects of numerical mesh generation. The approaches vary considerably. Roache states in the forward of reference [14], "The literature of boundary-fitted grid generation has suffered because the subject is too easy." His point is that mesh generation methods have usually been developed for application to a narrow set of geometries and that they generally fail when applied to other types of geometries.

One would like to generalize many of the mesh generation methods so that they become applicable to a wide variety of geometries. Castillo states in the preface of reference [14], "Many grid-generation algorithms have a continuum limit; that is, as the distance between gridpoints goes to zero, the limiting grid distribution becomes a transformation that is a solution to a boundary value problem for a partial differential equation or variational equation." This suggests that many algebraic mesh generation methods approximate the solutions of partial differential equations. There are many mesh generation methods which are directly based on partial differential equations. The most common of these are methods based on solving some numerical

representation of an elliptic partial differential equation, usually a Poisson equation. These partial differential methods can be shown to have an equivalent variational formulation. The approach of this work has been to base the development of the mesh equations on variational principles.

The next several sections discuss various aspects of the development of the mesh equations based on the calculus of variations. The development of the mesh speed equations is then discussed.

The Calculus of Variations and the Mesh Equations

The calculus of variations is concerned with finding the stationary value of an integral. For the generation of a two-dimensional mesh, a variational statement may be formulated as a double integral of the form

$$I = \int\int L\left(\vec{q}, \dot{\vec{q}}, \vec{t}\right)\, d\xi\, d\eta \tag{5.1}$$

where $\vec{q}$ is the vector of dependent (state) variables, $\dot{\vec{q}}$ is the vector of the rates of the state vector, and $\vec{t}$ is the vector of independent variables. For the two-dimensional mesh

$$\vec{q}^{\,T} = \vec{r}^{\,T} = [\, x,\, y\,] \tag{5.2}$$

and

$$\vec{t}^{\,T} = [\, \xi\,,\, \eta\,]. \tag{5.3}$$

The Euler-Lagrange equations, which are the necessary conditions for a stationary value of the integral, are

$$\frac{d}{dt_i}\left(\frac{\partial L}{\partial q_{j,i}}\right) - \frac{\partial L}{\partial q_j} = 0, \tag{5.4}$$

where $q_{j,i}$ denotes the differentiation

$$\frac{\partial q_j}{\partial t_i}. \tag{5.5}$$

The subscripts i and j denote the elements of $\vec{r}$ and $\vec{t}$ and repeated subscripts indicate summation. Thus, for the two-dimensional mesh, the Euler-Lagrange equations are

$$\left(\frac{\partial}{\partial \xi}\frac{\partial}{\partial x_\xi} + \frac{\partial}{\partial \eta}\frac{\partial}{\partial x_\eta} - \frac{\partial}{\partial x}\right) L = 0 \tag{5.6}$$

and

$$\left(\frac{\partial}{\partial \xi}\frac{\partial}{\partial y_\xi} + \frac{\partial}{\partial \eta}\frac{\partial}{\partial y_\eta} - \frac{\partial}{\partial y}\right) L = 0. \tag{5.7}$$

The major distinction among variational mesh generation methods is whether the method solves a numerical approximation of the integral equations (5.1) or the Euler-Lagrange equations (5.4). Methods which solve numerical approximation of the integral equations (5.4) are known as direct optimization methods. The present work is based on solving the numerical approximation of the Euler-Lagrange equations (5.4).

The Approach to Solution Adaptive Meshes

Another major disctinction among variational mesh generation methods is the manner in which the functional L is defined. It is desirable to generate a mesh such that the errors in the numerical solution are minimized. These are errors due to the discrete representation of the continuous problem. Developing an explicit expression for the errors as a function of the mesh point properties and the flow solution is difficult. The approach followed here is to assume a direct relation between the magnitude of the errors and some measurable qualities of the mesh.

One such measurable quality is the smoothness in the variation of the mesh spacing. The numerical approximations of derivatives are generally more accurate on smooth meshes.

Another measurable quality of the mesh is the orthogonality of mesh lines on the interior and on the boundary. The evaluation of cell-face fluxes are generally more accurate if the mesh lines of the finite-volume cells are not excessively skewed. The evaluation of boundary conditions often assume that mesh lines are normal to the boundaries.

Another measurable quality of the mesh is the variation of the volume of the finite-volume cells and the variation of solution gradients on the mesh. The consistency requirement of a numerical method states that as the size of the finite-volume cell decreases, the errors should decrease. Sources of significant errors are regions of high flow gradients. It is desirable then to decrease the sizes of the finite-volume cells in the regions of high flow gradients.

Brackbill and Saltzman [12] introduced a form for the functional L such that the smoothness, orthogonality, and volume adaption of the mesh was measured. Using this in the Euler-Lagrange equations, they were able to generate structured meshes with the desired qualities. They define the functional as the linear combination

$$L = L_S + L_O + L_A, \tag{5.8}$$

where for a two-dimensional mesh

$$L_S = \lambda_S \frac{\vec{r}_\xi \cdot \vec{r}_\xi + \vec{r}_\eta \cdot \vec{r}_\eta}{J}, \tag{5.9}$$

$$L_O = \lambda_O \left(\vec{r}_\xi \cdot \vec{r}_\eta\right)^2, \tag{5.10}$$

and

$$L_A = \lambda_A W J^2. \tag{5.11}$$

Thus the variational integral becomes the linear combination

$$I = I_S + I_O + I_A. \tag{5.12}$$

The λ_S, λ_O, and λ_A are constants which weigh the importance of each functional relative to the other functionals. One should note that there are differences in the dimensions of each functional. This makes the selection of λ_S, λ_O, and λ_A problem-dependent.

The smoothness integral, I_S, is a measure of the gradient in the mesh along each mesh coordinate line. A minimum is reached when the mesh is uniformly spaced along the mesh coordinate line. The orthogonality integral, I_O, is a measure of the cross product of the tangent vectors at a mesh point. The minimum is reached when the mesh is fully orthogonal. In this case, the minimum is zero. The volume adaption integral, I_A, is an equidistribution statement. It attempts to equidistribute the quantity $W J$. The weighting function W is usually directly related to the flow solution gradients. In regions of high solution gradients, W is large and so this forces J, which is essentially the volume of the cell, to be reduced.

The Mathematical Characteristics of the Integral Equations

Understanding the mathematical characteristics of the variational integral equations is important in developing a procedure for the numerical solution of the mesh equations.

Castillo [14] investigates the mathematical character of the smoothness and volume adaption integral equations. He states that the Euler-Lagrange equations for smoothness are a set of linear, uncoupled, elliptic partial differential equations. The maximum principle leads to a unique and correct solution. The Euler-Lagrange equations for volume adaption are a set of nonlinear, coupled, nonelliptic (mixed) partial differential equations that may not lead to a unique solution. The common approach is to combine the smoothness and volume adaption equations and rely on the uniqueness properties of the smoothness equations to pick out the smoothest, nonunique solution from the volume adaption equations.

The orthogonality equations also have nonunique solutions. The measure of orthogonality can not distinguish between a correctly orthogonal mesh line and one folded over on itself. Again, the common approach is to combine the orthogonality equations with the smoothness and volume adaption equations. The uniqueness properties of the smoothness equations lead to a unique solution of the combined equations.

The combined system of equations is a system of quasi-linear, coupled partial differential equations. The equations appear to be well-posed as long as orthogonality and volume adaption measures are not too large compared to the smoothness measure. This implies that there is a limit on the values of λ_O and λ_A.

Direct Optimization Methods

Direct optimization methods numerically approximate the integral equation (5.1) and then use nonlinear programming (optimization) to find the mesh which minimizes the integral. This approach is preferred to solving the Euler-Lagrange equations

because equation (5.1) is a simpler form.

Carcaillet [13] forms a discrete representation of equation (5.1) to form an objective function. The mesh is obtained through an unconstrained minimization of the objective function using the first-order conjugate-gradient, Fletcher-Reeves method. The method is applied to generate the surface and flowfield meshes over a wing/body geometry.

Castillo [14] warns of the possiblity that the discrete representation of equation (5.1) may exhibit strong decoupling problems due to the appearance of only first-order derivatives which when discretized do not involve the point being solved. Also, certain integral constraints may not be satisfied. He mentions that in the Euler-Lagrange representation, the decoupling problem does not exist and that the integral constraints are satisfied. The decoupling problem is removed in reference [13] through the definition of the discrete form of the functionals.

Methods Based on the Euler-Lagrange Equations

The alternative to the discretization of the integral equations (5.1) and the subsequent optimization procedure is to formulate and solve the Euler-Lagrange equations in the form of equations (5.4). This approach was presented by Brackbill and Saltzman [12] as a generalization of the Poisson equation formulations.

The form of the functional, L, used by Brackbill and Saltzman was presented in a previous section. This functional was inserted into the Euler-Lagrange equations and a system of second-order, quasi-linear partial differential equations for the mesh points, $\vec{r}$, resulted. They used second-order, central finite-differences to approximate the derivatives. The resulting algebraic system of equations was then solved using

the point-Jacobi iteration method. They apply the system of equations to compute solution adaptive meshes for two-dimensional domains and for adaption on three-dimensional surfaces. Brackbill and Saltzman do mention a limit on the values of λ_O and λ_A above which a convergent mesh solution was not possible. They do state that the values of the limits increase as the number of mesh points increases. They also suggest that the weighting function W be smoothed prior to use to help the convergence. The mesh solutions show a competition between the smoothness, orthogonality, and volume adaption qualities. Thus more clustering usually comes at the expense of orthogonality and smoothness in the mesh. It was also observed that orthogonality was more strongly controlled in the larger cells.

The Brackbill and Saltzman approach is used by Kreis et al. [33] for solution adaptive meshes on two-dimensional domains. They use the ADI method of Peacemann and Rachford to iteratively solve the system of equations. They also discuss scaling concerns of the equations with respect to the values of λ_O and λ_A.

Holcomb [31] discusses the implementation of the Brackbill and Saltzman system for the mesh adaption of three-dimensional flowfields about rocket plumes. He solves the system of mesh equations using a line-relaxation method. The mesh equations are iterated once every two time steps of the flow solver. The weighting function W is defined to adapt to inviscid and viscous flowfields. He notes that it is helpful to put a maximum limit on the value of W in addition to smoothing. He states that the Brackbill and Saltzman system has problems computing meshes for complex geometries.

Roach and Steinberg [39] present a modified version of the Brackbill and Saltzman system. They perform different discretizations of the Euler-Lagrange equations

that address problems with solving the system of equations for complicated geometries.

The variational approach and the Euler-Lagrange equations represent a minimization problem. References [16] and [17] discuss a series of successful, three-dimensional solution adaptive mesh methods that have been developed based on the minimization of the total energy of a system of linear and torsion displacement springs. The linear springs with variable stiffness coefficients control the spacing along coordinate lines in an equidistribution approach. The torsion springs control the orthogonality of the mesh lines. This approach is essentially a variational statement applied to a discrete system. It is suggested that this discrete approach removes truncation errors found in the discretization of the partial differential equations.

The work here involves solving the Euler-Lagrange equations. The mesh equations may be more complex than those resulting from direct optimization methods; however, it is felt that their solution may require less computational effort. Further, it is desired to have mesh equations expressed in terms of the mesh variables. This is because the approach for developing the mesh speed equation is to take the time differentiation of the mesh equations. The next section presents the form of the mesh equations used in this work.

Modifications to the Mesh Equations Due to Hindman

The nondimensionalization of the orthogonality and volume adaption integrals is desired for consistency with the smoothness integral. This is done through modifications proposed by Hindman [27]. The smoothness functional, L_S, remains the

same. The functionals are defined as

$$L_S = \lambda_S \frac{\vec{r}_\xi \cdot \vec{r}_\xi + \vec{r}_\eta \cdot \vec{r}_\eta}{J}, \tag{5.13}$$

$$L_O = \lambda_O \left(\frac{\vec{r}_\xi \cdot \vec{r}_\eta}{J} \right)^2, \tag{5.14}$$

and

$$L_A = \lambda_A \ (W \ J \ )^2 \ / \ K^2. \tag{5.15}$$

The K is an average value of WJ defined by

$$K = \frac{\int_V W \ dx \ dy}{(\xi max - \xi min) \ (\eta max - \eta min)}. \tag{5.16}$$

Numerically, K is evaluated as

$$K = \frac{\sum_{i=1}^{NI} \ \sum_{j=1}^{NJ} \ W_{i,j} \ V_{i,j}}{(NI-1) \ (NJ-1)}. \tag{5.17}$$

The expression for L_A essentially equidistributes WJ, where W is some measure of the solution to which the mesh adapts.

If these functionals are used in the Euler-Lagrange equations and the terms collected, the resulting equations are a system of quasi-linear, partial differential equations for the mesh law, which can be expressed in the form

$$G(\vec{r}) = A \ \vec{r}_{\xi\xi} + B \ \vec{r}_{\xi\eta} + C \ \vec{r}_{\eta\eta} + D \ \vec{r}_\xi + E \ \vec{r}_\eta = 0 \tag{5.18}$$

where

$$\vec{r}^{\,T} = [\ x, \ y \ ]$$

and

$$A = A_S + A_O + A_A,$$

$$B = B_S + B_O + B_A,$$

$$C = C_S + C_O + C_A,$$

$$D = D_S + D_O + D_A,$$

and

$$E = E_S + E_O + E_A.$$

The expressions for the smoothness matrices are

$$A_S = \frac{\lambda_S \, \alpha}{J^3} [S], \tag{5.19}$$

$$B_S = \frac{-2 \, \lambda_S \, \beta}{J^3} [S], \tag{5.20}$$

$$C_S = \frac{\lambda_S \, \gamma}{J^3} [S], \tag{5.21}$$

$$D_S = [0], \tag{5.22}$$

and

$$E_S = [0], \tag{5.23}$$

where

$$S = \begin{bmatrix} y_\xi^2 + y_\eta^2 & -(x_\xi y_\xi + x_\eta y_\eta) \\ -(x_\xi y_\xi + x_\eta y_\eta) & x_\xi^2 + x_\eta^2 \end{bmatrix} \tag{5.24}$$

or

$$S = \begin{bmatrix} \delta & -\epsilon \\ -\epsilon & \rho \end{bmatrix}. \tag{5.25}$$

The [0] is the null matrix. The other variables are defined as

$$J = x_\xi \, y_\eta - x_\eta \, y_\xi, \tag{5.26}$$

$$\alpha = x_\eta^2 + y_\eta^2, \tag{5.27}$$

$$\beta = x_\xi\, x_\eta + y_\xi\, y_\eta, \tag{5.28}$$

$$\gamma = x_\xi^2 + y_\xi^2, \tag{5.29}$$

$$\delta = y_\xi^2 + y_\eta^2, \tag{5.30}$$

$$\epsilon = x_\xi\, y_\xi + x_\eta\, y_\eta, \tag{5.31}$$

and

$$\rho = x_\xi^2 + x_\eta^2. \tag{5.32}$$

The expressions for the orthogonality matrices are

$$A_O = \frac{\lambda_O \alpha}{J^4} \begin{bmatrix} y_\xi(2\beta y_\eta + \alpha y_\xi) & -(\gamma x_\eta y_\eta + 2\alpha x_\xi y_\xi) \\ -(\gamma x_\eta y_\eta + 2\alpha x_\xi y_\xi) & x_\xi(2\beta x_\eta + \alpha x_\xi) \end{bmatrix} = \frac{\lambda_O \alpha}{J^4}[O1], \tag{5.33}$$

$$B_O = -\frac{\lambda_O(2\beta^2 + \alpha\gamma)}{J^4} \begin{bmatrix} 2y_\xi y_\eta & -\phi \\ -\phi & 2x_\xi x_\eta \end{bmatrix} = -\frac{\lambda_O(2\beta^2 + \alpha\gamma)}{J^4}[O2], \tag{5.34}$$

$$C_O = \frac{\lambda_O \gamma}{J^4} \begin{bmatrix} y_\eta(\gamma y_\eta + 2y_\xi \beta) & -(\alpha x_\xi y_\xi + 2\gamma x_\eta y_\eta) \\ -(\alpha x_\xi y_\xi + 2\gamma x_\eta y_\eta) & x_\eta(\gamma x_\eta + 2x_\xi \beta) \end{bmatrix} = \frac{\lambda_O \gamma}{J^4}[O3], \tag{5.35}$$

$$D_O = [0], \tag{5.36}$$

and

$$E_O = [0]. \tag{5.37}$$

The ϕ is defined as

$$\phi = x_\xi\, y_\eta + x_\eta\, y_\xi. \tag{5.38}$$

The expressions for the volume adaption matrices are

$$A_A = \frac{\lambda_A W^2}{K^2} \begin{bmatrix} y_\eta^2 & -x_\eta y_\eta \\ -x_\eta y_\eta & x_\eta^2 \end{bmatrix} = \frac{\lambda_A W^2}{K^2} [A1], \tag{5.39}$$

$$B_A = \frac{\lambda_A W^2}{K^2} \begin{bmatrix} -2y_\xi y_\eta & \phi \\ \phi & -2x_\xi x_\eta \end{bmatrix} = \frac{\lambda_A W^2}{K^2} [A2], \tag{5.40}$$

$$C_A = \frac{\lambda_A W^2}{K^2} \begin{bmatrix} y_\xi^2 & -x_\xi y_\xi \\ -x_\xi y_\xi & x_\xi^2 \end{bmatrix} = \frac{\lambda_A W^2}{K^2} [A3], \tag{5.41}$$

$$D_A = \frac{\lambda_A JWW_\eta}{K^2} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \tag{5.42}$$

and

$$E_A = \frac{\lambda_A JWW_\xi}{K^2} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{5.43}$$

Second-order central differences are used to approximate the derivatives. This is discussed in the next chapter along with the numerical method for solving the algebraic system of equations.

The Mesh Speed Equations

The objective of this work is to develop the mesh speed equations for the direct computation of the mesh speeds. The approach is to obtain the mesh speed equations

from the time differentiation of equation (5.18). The objective is to write the mesh speed equations in the form

$$G_\tau(x,y) = A^* \vec{z}_{\xi\xi} + B^* \vec{z}_{\xi\eta} + C^* \vec{z}_{\eta\eta} + D^* \vec{z}_\xi + E^* \vec{z}_\eta + \vec{T}^* = 0 \qquad (5.44)$$

where

$$\vec{z}^{\,T} = [\, x_\tau,\ y_\tau \,].$$

Differentiating the mesh equation using the chain rule results in the expression

$$\begin{aligned} G_\tau(x,y) &= A\ \vec{z}_{\xi\xi} + B\ \vec{z}_{\xi\eta} + C\ \vec{z}_{\eta\eta} + D\ \vec{z}_\xi + E\ \vec{z}_\eta \\ &+ A_\tau\ \vec{r}_{\xi\xi} + B_\tau\ \vec{r}_{\xi\eta} + C_\tau\ \vec{r}_{\eta\eta} + D_\tau\ \vec{r}_\xi + E_\tau\ \vec{r}_\eta = 0. \end{aligned} \qquad (5.45)$$

Note that the differentiations of ξ, η, and τ can be interchanged due to the independence between the coordinates.

The time differentiations A_τ, B_τ, C_τ, D_τ, and E_τ must be performed. Consider A_τ,

$$A_\tau = (A_S)_\tau + (A_O)_\tau + (A_A)_\tau. \qquad (5.46)$$

Similar expressions exist for B_τ, C_τ, D_τ, and E_τ.

The time differentiation of the smoothness matrices result in the following expressions:

$$(A_S)_\tau = \frac{\lambda_S\ \alpha_\tau}{J^3}\ [S] - \frac{3\ \lambda_S\ \alpha\ J_\tau}{J^4}\ [S] + \frac{\lambda_S\ \alpha}{J^3}\ [S]_\tau, \qquad (5.47)$$

$$(B_S)_\tau = -\frac{2\ \lambda_S\ \beta_\tau}{J^3}\ [S] + \frac{6\ \lambda_S\ \beta\ J_\tau}{J^4}\ [S] - \frac{2\ \lambda_S\ \beta}{J^3}\ [S]_\tau, \qquad (5.48)$$

$$(C_S)_\tau = \frac{\lambda_S\ \gamma_\tau}{J^3}\ [S] - \frac{3\ \lambda_S\ \gamma\ J_\tau}{J^4}\ [S] + \frac{\lambda_S\ \gamma}{J^3}\ [S]_\tau, \qquad (5.49)$$

$$(D_S)_\tau = [0], \qquad (5.50)$$

and

$$(E_S)_\tau = [0], \tag{5.51}$$

where

$$\alpha_\tau \;=\; 2\, x_\eta\, (x_\tau)_\eta \;+\; 2\, y_\eta\, (y_\tau)_\eta \;=\; 2\, \vec{r}_\eta \cdot \vec{z}_\eta, \tag{5.52}$$

$$\beta_\tau \;=\; x_\eta\, (x_\tau)_\xi \;+\; x_\xi\, (x_\tau)_\eta \;+\; y_\eta\, (y_\tau)_\xi \;+\; y_\xi\, (y_\tau)_\eta \;=\; \vec{r}_\eta \cdot \vec{z}_\xi \;+\; \vec{r}_\xi \cdot \vec{z}_\eta, \tag{5.53}$$

$$\gamma_\tau \;=\; 2\, x_\xi\, (x_\tau)_\xi \;+\; 2\, y_\xi\, (y_\tau)_\xi \;=\; 2\, \vec{r}_\xi \cdot \vec{z}_\xi, \tag{5.54}$$

$$J_\tau \;=\; y_\eta\, (x_\tau)_\xi \;+\; x_\xi\, (y_\tau)_\eta \;-\; y_\xi\, (x_\tau)_\eta \;-\; x_\eta\, (y_\tau)_\xi, \tag{5.55}$$

$$[S]_\tau = \begin{bmatrix} \delta_\tau & -\epsilon_\tau \\ -\epsilon_\tau & \rho_\tau \end{bmatrix}, \tag{5.56}$$

$$\delta_\tau \;=\; 2\, y_\xi\, (y_\tau)_\xi \;+\; 2\, y_\eta\, (y_\tau)_\eta, \tag{5.57}$$

$$\epsilon_\tau \;=\; y_\xi\, (x_\tau)_\xi \;+\; x_\xi\, (y_\tau)_\xi \;+\; y_\eta\, (x_\tau)_\eta \;+\; x_\eta\, (y_\tau)_\eta, \tag{5.58}$$

and

$$\rho_\tau \;=\; 2\, x_\xi\, (x_\tau)_\xi \;+\; 2\, x_\eta\, (x_\tau)_\eta. \tag{5.59}$$

The time differentiation of orthogonality matrices results in the following expressions:

$$(A_O)_\tau \;=\; \frac{\lambda_O\, \alpha_\tau}{J^4}\, [O1] \;-\; \frac{4\, \lambda_O\, \alpha\, J_\tau}{J^5}\, [O1] \;+\; \frac{\lambda_O\, \alpha}{J^4}\, [O1]_\tau, \tag{5.60}$$

$$\begin{aligned} (B_O)_\tau \;=\; & -\frac{4\, \lambda_O\, \beta\, \beta_\tau}{J^4}\, [O2] \;-\; \frac{\lambda_O\, \gamma\, \alpha_\tau}{J^4}\, [O2] \;-\; \frac{\lambda_O\, \alpha\, \gamma_\tau}{J^4}\, [O2] \\ & +\; \frac{4\, \lambda_O \left(2\, \beta^2 \;+\; \alpha\, \gamma \right)}{J^5}\, J_\tau\, [O2] \\ & -\; \frac{\lambda_O \left(2\, \beta^2 \;+\; \alpha\, \gamma \right)}{J^4}\, [O2]_\tau, \end{aligned} \tag{5.61}$$

$$(C_O)_\tau = \frac{\lambda_O \gamma_\tau}{J^4} [O3] - \frac{4 \lambda_O \gamma J_\tau}{J^5} [O3] + \frac{\lambda_O \gamma}{J^4} [O3]_\tau \tag{5.62}$$

$$\phi_\tau = y_\eta (x_\tau)_\xi + x_\xi (y_\tau)_\eta + y_\xi (x_\tau)_\eta + x_\eta (y_\tau)_\xi, \tag{5.63}$$

$$(D_O)_\tau = [0], \tag{5.64}$$

and

$$(E_O)_\tau = [0]. \tag{5.65}$$

The coefficient matrices for the mesh speeds can be presented by using some shorthand notation:

$$OC1 = \lambda_O / J^4,$$

$$OC2 = \lambda_O \alpha / J^4 = (OC1)\, \alpha,$$

$$OC3 = -\lambda_O \left(2\beta^2 + \alpha\gamma \right) / J^4 = -(OC1) \left(2\beta^2 + \alpha\gamma \right),$$

$$OC4 = \lambda_O \gamma / J^4 = (OC1)\, \gamma,$$

$$OC5 = -4 \lambda_O \alpha / J^5 = -4\, (OC2) / J,$$

$$OC6 = -4 \lambda_O \beta / J^4 = -4\, (OC1)\, \beta,$$

$$OC7 = -\lambda_O \gamma / J^4 = -(OC4),$$

$$OC8 = -\lambda_O \alpha / J^4 = -(OC2),$$

$$OC9 = 4 \lambda_O \left(2\beta^2 + \alpha\gamma \right) / J^5 = -4\, (OC3) / J,$$

$$OC10 = -4 \lambda_O \gamma / J^5 = -4\, (OC4) / J,$$

$$[O1]\, \vec{r}_{\xi\xi} = \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix},$$

$$[O2]\ \vec{r}_{\xi\eta} \ = \ \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\},$$

and

$$[O3]\ \vec{r}_{\eta\eta} \ = \ \left\{ \begin{array}{c} c_1 \\ c_2 \end{array} \right\}.$$

Thus,

$$(A_O)_\tau\ \vec{r}_{\xi\xi} = (OC1) \left\{ \begin{array}{c} a_1 \\ a_2 \end{array} \right\} \alpha_\tau + (OC5) \left\{ \begin{array}{c} a_1 \\ a_2 \end{array} \right\} J_\tau + (OC2)\,[O1]_\tau\,\vec{r}_{\xi\xi}, \quad (5.66)$$

$$\begin{aligned} (B_O)_\tau\ \vec{r}_{\eta\eta} \ &= \ (OC6) \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\} \beta_\tau + (OC7) \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\} \alpha_\tau + (OC8) \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\} \gamma_\tau \\ &+ \ (OC9) \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\} J_\tau + (OC3)\,[O2]_\tau\,\vec{r}_{\xi\eta}, \end{aligned} \quad (5.67)$$

and

$$(C_O)_\tau\ \vec{r}_{\eta\eta} = (OC1) \left\{ \begin{array}{c} c_1 \\ c_2 \end{array} \right\} \gamma_\tau + (OC10) \left\{ \begin{array}{c} c_1 \\ c_2 \end{array} \right\} J_\tau + (OC4)\,[O3]_\tau\,\vec{r}_{\eta\eta}. \quad (5.68)$$

Expanding the terms and writing as coefficients of the mesh speeds result in the expressions:

$$\begin{aligned} (A_O)_\tau\ \vec{r}_{\xi\xi} \ &= \ (OC5)\ y_\eta \left\{ \begin{array}{c} a_1 \\ a_2 \end{array} \right\} (x_\tau)_\xi \\ &- \ (OC5)\ x_\eta \left\{ \begin{array}{c} a_1 \\ a_2 \end{array} \right\} (y_\tau)_\xi \\ &+ \ \left[\, 2\,(OC1)\ x_\eta \ - \ (OC5)\ y_\xi \,\right] \left\{ \begin{array}{c} a_1 \\ a_2 \end{array} \right\} (x_\tau)_\eta \end{aligned}$$

$$
\begin{aligned}
&+ \left[\, 2\,(OC1)\, y_\eta \,+\, (OC5)\, x_\xi \,\right] \left\{ \begin{array}{c} a_1 \\ a_2 \end{array} \right\} (y_\tau)_\eta \\
&+ (OC2) \left[\begin{array}{c} (e1a)x_{\xi\xi} + (e2a)y_{\xi\xi} \\ (e2a)x_{\xi\xi} + (e3a)y_{\xi\xi} \end{array} \right] (x_\tau)_\xi \\
&+ (OC2) \left[\begin{array}{c} (e1b)x_{\xi\xi} + (e2b)y_{\xi\xi} \\ (e2b)x_{\xi\xi} + (e3b)y_{\xi\xi} \end{array} \right] (y_\tau)_\xi \\
&+ (OC2) \left[\begin{array}{c} (e1c)x_{\xi\xi} + (e2c)y_{\xi\xi} \\ (e2c)x_{\xi\xi} + (e3c)y_{\xi\xi} \end{array} \right] (x_\tau)_\eta \\
&+ (OC2) \left[\begin{array}{c} (e1d)x_{\xi\xi} + (e2d)y_{\xi\xi} \\ (e2d)x_{\xi\xi} + (e3d)y_{\xi\xi} \end{array} \right] (y_\tau)_\eta, \qquad (5.69)
\end{aligned}
$$

$$
\begin{aligned}
(B_O)_\tau\, \vec{r}_{\xi\eta} \;=\; & \left[\, (OC6)\, x_\eta \,+\, 2\,(OC8)\, x_\xi \,+\, (OC9)\, y_\eta \,\right] \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\} (x_\tau)_\xi \\
&+ \left[\, (OC6)\, y_\eta \,+\, 2\,(OC8)\, y_\xi \,-\, (OC9)\, x_\eta \,\right] \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\} (y_\tau)_\xi \\
&+ \left[\, (OC6)\, x_\xi \,+\, 2\,(OC7)\, x_\eta \,-\, (OC9)\, y_\xi \,\right] \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\} (x_\tau)_\eta \\
&+ \left[\, (OC6)\, y_\xi \,+\, 2\,(OC7)\, y_\eta \,+\, (OC9)\, x_\xi \,\right] \left\{ \begin{array}{c} b_1 \\ b_2 \end{array} \right\} (y_\tau)_\eta \\
&+ (OC3) \left[\begin{array}{c} -y_\eta y_{\xi\eta} \\ 2x_\eta y_{\xi\eta} - y_\eta x_{\xi\eta} \end{array} \right] (x_\tau)_\xi \\
&+ (OC3) \left[\begin{array}{c} 2y_\eta x_{\xi\eta} - x_\eta y_{\xi\eta} \\ -x_\eta x_{\xi\eta} \end{array} \right] (y_\tau)_\xi
\end{aligned}
$$

$$+ \quad (OC3) \begin{bmatrix} -y_\xi y_{\xi\eta} \\ 2x_\xi y_{\xi\eta} - y_\xi x_{\xi\eta} \end{bmatrix} (x_\tau)_\eta$$

$$+ \quad (OC3) \begin{bmatrix} 2y_\xi x_{\xi\eta} - x_\xi y_{\xi\eta} \\ -x_\xi x_{\xi\eta} \end{bmatrix} (y_\tau)_\eta, \tag{5.70}$$

and

$$\begin{aligned}
(C_O)_\tau \, \vec{r}_{\eta\eta} \;=\; & \left[2\,(OC1)\, x_\xi \;+\; (OC10)\, y_\eta \right] \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} (x_\tau)_\xi \\
& + \left[2\,(OC1)\, y_\xi \;-\; (OC10)\, x_\eta \right] \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} (y_\tau)_\xi \\
& - \; (OC10)\, y_\xi \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} (x_\tau)_\eta \\
& + \; (OC10)\, x_\xi \begin{Bmatrix} c_1 \\ c_2 \end{Bmatrix} (y_\tau)_\eta \\
& + \; (OC4) \begin{bmatrix} (f1a)x_{\eta\eta} + (f2a)y_{\eta\eta} \\ (f2a)x_{\eta\eta} + (f3a)y_{\eta\eta} \end{bmatrix} (x_\tau)_\xi \\
& + \; (OC4) \begin{bmatrix} (f1b)x_{\eta\eta} + (f2b)y_{\eta\eta} \\ (f2b)x_{\eta\eta} + (f3b)y_{\eta\eta} \end{bmatrix} (y_\tau)_\xi \\
& + \; (OC4) \begin{bmatrix} (f1c)x_{\eta\eta} + (f2c)y_{\eta\eta} \\ (f2c)x_{\eta\eta} + (f3c)y_{\eta\eta} \end{bmatrix} (x_\tau)_\eta \\
& + \; (OC4) \begin{bmatrix} (f1d)x_{\eta\eta} + (f2d)y_{\eta\eta} \\ (f2d)x_{\eta\eta} + (f3d)y_{\eta\eta} \end{bmatrix} (y_\tau)_\eta,
\end{aligned} \tag{5.71}$$

where

$$e1a \;=\; 2\, x_\eta \, y_\xi \, y_\eta,$$

$$e1b = 2 \left(y_\eta \beta + y_\xi y_\eta^2 + \alpha y_\xi \right),$$

$$e1c = 2 \left(x_\xi y_\eta + x_\eta y_\xi \right) y_\xi,$$

$$e1d = 2 \left(\beta + 2 y_\xi y_\eta \right) y_\xi,$$

$$e2a = -2 \left(x_\xi x_\eta y_\eta + \alpha y_\xi \right),$$

$$e2b = -2 \left(x_\eta y_\xi y_\eta + \alpha x_\xi \right),$$

$$e2c = -\left(\gamma y_\eta + 4 x_\xi x_\eta y_\xi \right),$$

$$e2d = -\left(\gamma x_\eta + 4 x_\xi y_\xi y_\eta \right),$$

$$e3a = 2 \left(x_\eta \beta + x_\xi x_\eta^2 + \alpha x_\xi \right),$$

$$e3b = 2 x_\xi x_\eta y_\eta,$$

$$e3c = 2 \left(\beta + 2 x_\xi x_\eta \right) x_\xi,$$

$$e3d = 2 \left(y_\xi y_\eta + x_\xi y_\eta \right) x_\xi,$$

$$f1a = 2 \left(x_\eta y_\xi + x_\xi y_\eta \right) y_\eta,$$

$$f1b = 2 \left(\beta + 2 y_\xi y_\eta \right) y_\eta,$$

$$f1c = 2 x_\xi y_\xi y_\eta,$$

$$f1d = 2 \left(y_\xi \beta + y_\xi^2 y_\eta + \gamma y_\eta \right),$$

$$f2a = -\left(\alpha y_\xi + 4 x_\xi x_\eta y_\eta \right),$$

$$f2b = -\left(\alpha x_\xi + 4 x_\eta y_\xi y_\eta \right),$$

$$f2c = -2 \left(x_\xi x_\eta y_\xi + \gamma y_\eta \right),$$

$$f2d = -2 \left(x_\xi y_\xi y_\eta + \gamma x_\eta \right),$$

$$f3a \;=\; 2\,\left(\,\beta \;+\; 2\,x_\xi\,x_\eta\,\right)\,x_\eta,$$

$$f3b \;=\; 2\,\left(\,x_\xi\,y_\eta \;+\; y_\xi\,x_\eta\,\right)\,x_\eta,$$

$$f3c \;=\; 2\,\left(\,x_\xi\,\beta \;+\; x_\xi^2\,x_\eta \;+\; \gamma\,x_\eta\,\right),$$

and

$$f3d \;=\; 2\,x_\xi\,x_\eta\,y_\xi.$$

The time differentiation of the volume adaption matrices results in the following expressions:

$$(A_A)_\tau \;=\; \frac{\lambda_A}{K^2}\,\left\{\,2\,W\,W_\tau\,\left[A1\right] \;+\; W^2\,\left[\,A1\,\right]_\tau\,\right\}, \tag{5.72}$$

where

$$\left[A1\right]_\tau = \begin{bmatrix} 2y_\eta(y_\tau)_\eta & -y_\eta(x_\tau)_\eta - x_\eta(y_\tau)_\eta \\ -y_\eta(x_\tau)_\eta - x_\eta(y_\tau)_\eta & 2x_\eta(x_\tau)_\eta \end{bmatrix}, \tag{5.73}$$

$$(B_A)_\tau = \frac{\lambda_A}{K^2}\left\{2WW_\tau\left[A2\right] + W^2\left[A2\right]_\tau\right\} \tag{5.74}$$

where

$$\left[A2\right]_\tau = \begin{bmatrix} -2[y_\eta(y_\tau)_\xi + y_\xi(y_\tau)_\eta] & \phi_\tau \\ \phi_\tau & -2[x_\eta(x_\tau)_\xi + x_\xi(x_\tau)_\eta] \end{bmatrix}, \tag{5.75}$$

$$(C_A)_\tau = \frac{\lambda_A}{K^2}\left\{2WW_\tau\left[A3\right] + W^2\left[A3\right]_\tau\right\} \tag{5.76}$$

where

$$\left[A3\right]_\tau = \begin{bmatrix} 2y_\xi(y_\tau)_\xi & -y_\xi(x_\tau)_\xi - x_\xi(y_\tau)_\xi \\ -y_\xi(x_\tau)_\xi - x_\xi(y_\tau)_\xi & 2x_\xi(x_\tau)_\xi \end{bmatrix}, \tag{5.77}$$

$$(D_A)_\tau = \frac{\lambda_A}{K^2}\left\{J_\tau W W_\eta + J W_\tau W_\eta + J W (W_\tau)_\eta\right\}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \tag{5.78}$$

and

$$(E_A)_\tau = \frac{\lambda_A}{K^2}\left\{J_\tau W W_\xi + J W_\tau W_\xi + J W (W_\tau)_\xi\right\}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{5.79}$$

Note that the expressions for A_τ, B_τ, C_τ, D_τ, and E_τ do not contain second partials of x_τ and y_τ; therefore

$$[A^*] = [A],$$

$$[B^*] = [B],$$

and

$$[C^*] = [C].$$

The task now is to rewrite the terms

$$A_\tau\, \vec{r}_{\xi\xi} \;+\; B_\tau\, \vec{r}_{\xi\eta} \;+\; C_\tau\, \vec{r}_{\eta\eta} \;+\; D_\tau\, \vec{r}_\xi \;+\; E_\tau\, \vec{r}_\eta \;+\; D\vec{z}_\xi \;+\; E\vec{z}_\eta \tag{5.80}$$

into the form of

$$D^*\, \vec{z}_\xi \;+\; E^*\, \vec{z}_\eta \;+\; \vec{T}^*. \tag{5.81}$$

This is done directly by performing the matrix-vector multiplications, forming the algebraic expressions, and then factoring out $\vec{z}_\xi$ and $\vec{z}_\eta$.

Once D^*, E^*, and $\vec{T}^*$ are formed, central differences can be used to approximate $\vec{z}_{\xi\xi}$, $\vec{z}_{\xi\eta}$, $\vec{z}_{\eta\eta}$, $\vec{z}_\xi$, and $\vec{z}_\eta$ and the other derivatives, and the system of algebraic equations can be solved. This is discussed in the next chapter.

The Form of the Weighting Function W

The volume adaption functional attempts to equidistribute the value of WJ throughout the mesh. If $W = 1$, the procedure will equidistribute J, which is essentially the volume of the finite-volume cell. For solution adaptive meshes, W is related to the gradients in the flow solution.

For the present work, W is defined in the form

$$W = \lambda_0 + \lambda_1 \left|\nabla f(U)\right| + \lambda_2 \exp\left(\lambda_3 \ \left|\nabla f(U)\right| \right). \tag{5.82}$$

For all of the computations, $\lambda_0 = 1.0$. This forced the mesh to form finite-volume cells of equal volume when the flow gradients were small. The $\nabla f(U)$ indicates a gradient of a function, f, of the flow solution. For inviscid flows, one choice of f is the density. For viscous flows, one choice of f is the Mach number.

One choice of the gradient is to use the gradient with respect to the computational coordinates ξ and η,

$$\left|\nabla_{i,j}\ f\right| = \left[\left(\frac{\partial f}{\partial \xi}\right)^2 + \left(\frac{\partial f}{\partial \eta}\right)^2\right]_{i,j}^{1/2}. \tag{5.83}$$

Central differences can be used to efficiently evaluate this gradient

$$\left|\nabla_{i,j}\ f\right| = \left\{\left[\frac{1}{2}(f_{i+1,j} - f_{i-1,j})\right]^2 + \left[\frac{1}{2}(f_{i,j+1} - f_{i,j-1})\right]^2\right\}_{i,j}^{1/2}. \tag{5.84}$$

The derivatives of W can be determined by applying the chain rule. Thus,

$$W_\xi \ = \ \Upsilon \ \left|\nabla f\right|_\xi, \tag{5.85}$$

$$W_\eta \ = \ \Upsilon \ \left|\nabla f\right|_\eta, \tag{5.86}$$

$$W_\tau \ = \ \Upsilon \ \left|\nabla f\right|_\tau. \tag{5.87}$$

$$W_{\xi\tau} = \Upsilon \, |\nabla f|_{\xi\tau} + \Phi \, |\nabla f|_{\xi}, \tag{5.88}$$

and

$$W_{\eta\tau} = \Upsilon \, |\nabla f|_{\eta\tau} + \Phi \, |\nabla f|_{\eta}, \tag{5.89}$$

where

$$\Upsilon = \lambda_1 + \lambda_2 \, \lambda_3 \exp(\lambda_3 \, |\nabla f|) \tag{5.90}$$

and

$$\Phi = \lambda_2 \, \lambda_3^2 \, |\nabla f|_{\tau} \exp(\lambda_3 \, |\nabla f|). \tag{5.91}$$

The spatial derivatives are approximated with second-order, central differences. The temporal derivatives are approximated with first-order, backward time differences.

The W is scaled through the following normalization:

$$|\nabla f| = \frac{|\nabla f| - |\nabla f|_{min}}{|\nabla f|_{max} - |\nabla f|_{min}}, \tag{5.92}$$

where $|\nabla f|_{min}$ and $|\nabla f|_{max}$ are specified minimum and maximums, respectively.

Another option is to use the gradient with respect to the physical coordinates x and y,

$$\left|\nabla_{i,j} \, f\right| = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2 \right]_{i,j}^{1/2}. \tag{5.93}$$

The computation of this gradient is more computationally expensive. In applications of this gradient, it was found that greater clustering was achieved, but that it seemed more prone to cause the mesh equations solution procedure to fail. The derivatives of W for this case were computed using second-order spatial finite-differences and first-order temporal finite-differences.

The finite differences of f performed in computing W assume f is a smooth function. This is not the case if there is a discontinuity in f. To prevent difficulties

in the finite-difference approximations, the function f may be smoothed prior to computing W and its derivatives. An elliptic smoothing is performed of the form

$$f^{s+1} = \left(f_{i-1,j} + f_{i+1,j} + 4\, f_{i,j} + f_{i,j-1} + f_{i,j+1} \right)^{s} / 8, \qquad (5.94)$$

where s denotes the smoothing pass. The computations presented in Chapters 9 and 10 use from 0 to 30 smoothing passes. This smoothing is symmetric with respect to point (i, j).

CHAPTER 6. THE NUMERICAL PROCEDURES FOR SOLVING THE MESH AND MESH SPEED EQUATIONS

This chapter discusses the numerical solution procedure for the mesh and mesh speed equations. The next chapter discusses the numerical procedures for the fluid dynamic equations. The following chapter then discusses the coupling of these equations for a dynamically adaptive mesh method.

The Numerical Solution of the Mesh Equation

The mesh equations were presented in equation (5.18) as

$$G(\vec{r}) = A\,\vec{r}_{\xi\xi} + B\,\vec{r}_{\xi\eta} + C\,\vec{r}_{\eta\eta} + D\,\vec{r}_{\xi} + E\,\vec{r}_{\eta} = 0 \tag{6.1}$$

where

$$\vec{r}^{\,T} = [\,x,\ y\,].$$

Second-order central differences are used to approximate the derivatives. If $\Delta\xi = 1$ and $\Delta\eta = 1$, the discretized mesh equation becomes

$$\begin{aligned} & A\,(\vec{r}_{i+1,j} - 2\vec{r}_{i,j} + \vec{r}_{i-1,j}) \\ + \ & \frac{1}{4}\,B\,(\vec{r}_{i+1,j+1} - \vec{r}_{i+1,j-1} - \vec{r}_{i-1,j+1} + \vec{r}_{i-1,j-1}) \\ + \ & C\,(\vec{r}_{i,j+1} - 2\vec{r}_{i,j} + \vec{r}_{i,j-1}) \\ + \ & \frac{1}{2}\,D\,(\vec{r}_{i+1,j} - \vec{r}_{i-1,j}) + \frac{1}{2}\,E\,(\vec{r}_{i,j+1} - \vec{r}_{i,j-1}) = 0. \end{aligned} \tag{6.2}$$

The system can be solved using a point Gauss-Seidel iteration method. The iteration equation is written for sweeps moving pointwise along constant-η lines from $i = 1$ to $i = NI$, and along constant-ξ lines from $j = 1$ to $j = NJ$ is

$$\begin{aligned}
\vec{r}_{i,j}^{\,m+1} &= \frac{1}{2}\,(A+C)^{-1} \\
&\Big[\ \frac{1}{4}\,B\,(\vec{r}_{i+1,j+1}^{\,m} - \vec{r}_{i+1,j-1}^{\,m+1} - \vec{r}_{i-1,j+1}^{\,m} + \vec{r}_{i-1,j-1}^{\,m+1}) \\
&+\ A\,(\vec{r}_{i+1,j}^{\,m} + \vec{r}_{i-1,j}^{\,m+1}) + C\,(\vec{r}_{i,j+1}^{\,m} + \vec{r}_{i,j-1}^{\,m+1}) \\
&+\ \frac{1}{2}\,D\,(\vec{r}_{i+1,j}^{\,m} - \vec{r}_{i-1,j}^{\,m+1}) + \frac{1}{2}\,E\,(\vec{r}_{i,j+1}^{\,m} - \vec{r}_{i,j-1}^{\,m+1})\ \Big]. \qquad (6.3)
\end{aligned}$$

The m is the iteration index. Relaxation is performed in the manner of

$$\vec{r}_{i,j}^{\,m+1} = \vec{r}_{i,j}^{\,m} + \omega_G \left(\vec{r}_{i,j}^{\,m+1} - \vec{r}_{i,j}^{\,m} \right). \qquad (6.4)$$

Other solution methods were tested such as the line relaxation and Newton's method. The Newton's method was found to be almost equivalent to the point relaxation approach. The line relaxation method did converge faster; however, it required more computational effort to reach the same level of convergence as the point relaxation method. The point Gauss-Seidel relaxation method is not vectorizable; however, the improvement in convergence over the point Jacobi, which is vectorizable, seems to outweigh any benefit from vectorization.

For Dirichlet boundary conditions, the location of the boundary mesh point is specified. The point relaxation procedure requires no modifications at the domain boundary.

Neumann boundary conditions are applied to impose orthogonality of the mesh lines at the boundaries. The orthogonality condition is

$$\vec{r}_\xi \cdot \vec{r}_\eta = 0 \qquad (6.5)$$

or

$$x_\xi \; x_\eta + y_\xi \; y_\eta = 0. \tag{6.6}$$

The boundary is parameterized with respect to the arclength along the boundary, s. For the constant-ξ boundaries, the metrics are

$$x_\xi = x_s \; s_\xi \quad \text{and} \quad y_\xi = y_s \; s_\xi.$$

Thus the orthogonality condition becomes

$$x_s \; x_\eta + y_s \; y_\eta = 0. \tag{6.7}$$

The derivatives with respect to η can be approximated with one-sided differences. The linearization of the boundary mesh points with respect to the previous iteration is

$$\vec{r}_{i,1}^{\,m+1} = \vec{r}_{i,1}^{\,m} + (\vec{r}_s)_{i,1}^{m} \;\; (\Delta s)_{i,1}^{m} \tag{6.8}$$

and

$$\vec{r}_{i,NJ}^{\,m+1} = \vec{r}_{i,NJ}^{\,m} + (\vec{r}_s)_{i,NJ}^{m} \;\; (\Delta s)_{i,NJ}^{m}. \tag{6.9}$$

These are substituted into the differenced orthogonality relation and one can then solve for Δs at the boundaries,

$$(\Delta s)_{i,1}^{m} = \frac{1}{3 \; (\vec{r}_s \cdot \vec{r}_s)_{i,1}^{m}} \; (\vec{r}_s)_{i,1}^{m} \cdot \left[\; 4\vec{r}_{i,2} - 3\vec{r}_{i,1} - \vec{r}_{i,3} \; \right]^m \tag{6.10}$$

and

$$(\Delta s)_{i,NJ}^{m} = \frac{1}{3 \; (\vec{r}_s \cdot \vec{r}_s)_{i,NJ}^{m}} \; (\vec{r}_s)_{i,NJ}^{m} \cdot \left[\; 4\vec{r}_{i,NJ-1} - 3\vec{r}_{i,NJ} - \vec{r}_{i,NJ-2} \; \right]^m . \tag{6.11}$$

Thus

$$s^{m+1} = s^m + \omega_{GB} \; (\Delta s)^m. \tag{6.12}$$

The new boundary mesh point can then be determined from

$$\vec{r}^{\,m+1} = \vec{r}_{\text{fit}} \left(s^{m+1} \right) \tag{6.13}$$

where $\vec{r}_{\text{fit}}$ is the curve fit of the boundary with respect to the parameter s.

For the constant-η boundaries the metrics are

$$x_\eta = x_s \; s_\eta \quad \text{and} \quad y_\eta = y_s \; s_\eta.$$

Thus the orthogonality condition becomes

$$x_s \; x_\xi + y_s \; y_\xi = 0. \tag{6.14}$$

The derivatives with respect to ξ can be approximated with one-sided differences. The linearizations of the boundary mesh point coordinates with respect to the previous iteration are

$$\vec{r}_{1,j}^{\,m+1} = \vec{r}_{1,j}^{\,m} + (\vec{r}_s)_{1,j}^{m} \; (\Delta s)_{1,j}^{m} \tag{6.15}$$

and

$$\vec{r}_{NI,j}^{\,m+1} = \vec{r}_{NI,j}^{\,m} + (\vec{r}_s)_{NI,j}^{m} \; (\Delta s)_{NI,j}^{m}. \tag{6.16}$$

These are substituted into the differenced orthogonality relation and one can then solve for Δs at the boundaries,

$$(\Delta s)_{1,j}^{m} = \frac{1}{3 \; (\vec{r}_s \cdot \vec{r}_s)_{1,j}^{m}} \; (\vec{r}_s)_{1,j}^{m} \cdot \left[\; 4\vec{r}_{2,j} - 3\vec{r}_{1,j} - \vec{r}_{3,j} \; \right]^m \tag{6.17}$$

and

$$(\Delta s)_{NI,j}^{m} = \frac{1}{3 \; (\vec{r}_s \cdot \vec{r}_s)_{NI,j}^{m}} \; (\vec{r}_s)_{NI,j} \cdot \left[\; 4\vec{r}_{NI-1,j} - 3\vec{r}_{NI,j} - \vec{r}_{NI-2,j} \; \right]^m. \tag{6.18}$$

The Numerical Solution of the Mesh Speed Equation

Second-order central differences are used to approximate the derivatives in the mesh speed equation. The point Gauss-Seidel iteration method is also used for the mesh speed equation in the same manner as for the mesh equation. The iteration equation is

$$\begin{aligned}
\vec{z}_{i,j}^{\,m+1} &= \frac{1}{2}\,(A+C)^{-1} \\
&\Big[\ \frac{1}{4}\,B\,(\vec{z}_{i+1,j+1}^{\,m} - \vec{z}_{i+1,j-1}^{\,m+1} - \vec{z}_{i-1,j+1}^{\,m} + \vec{z}_{i-1,j-1}^{\,m+1}) \\
&+\ A\,(\vec{z}_{i+1,j}^{\,m} + \vec{z}_{i-1,j}^{\,m+1}) + C\,(\vec{z}_{i,j+1}^{\,m} + \vec{z}_{i,j-1}^{\,m+1}) \\
&+\ \frac{1}{2}\,D\,(\vec{z}_{i+1,j}^{\,m} - \vec{z}_{i-1,j}^{\,m+1}) + \frac{1}{2}\,E\,(\vec{z}_{i,j+1}^{\,m} - \vec{z}_{i,j-1}^{\,m+1}) + \vec{T}\ \Big]. \qquad (6.19)
\end{aligned}$$

The relaxation is performed in the manner of

$$\vec{z}_{i,j}^{\,m+1} = \vec{z}_{i,j}^{\,m} + \omega_{GS}\left(\vec{z}_{i,j}^{\,m+1} - \vec{z}_{i,j}^{\,m}\right). \qquad (6.20)$$

For Dirichlet boundary conditions, the boundary values of the mesh speeds are known. The Neumann boundary conditions for the mesh speed law impose the orthogonality condition of the mesh lines at the boundary. The conditions are derived by starting with the orthogonality condition for the mesh,

$$\vec{r}_\xi \cdot \vec{r}_\eta = 0 \qquad (6.21)$$

and taking the time differentiation,

$$\frac{\partial}{\partial \tau}\left(\vec{r}_\xi \cdot \vec{r}_\eta\right) = 0 \qquad (6.22)$$

or

$$\vec{z}_\xi \cdot \vec{r}_\eta + \vec{r}_\xi \cdot \vec{z}_\eta = 0. \qquad (6.23)$$

Consider a boundary along a constant-η coordinate line and write

$$\vec{z}_\xi = \vec{z}_s \; s_\xi \quad \text{and} \quad \vec{r}_\xi = \vec{r}_s \; s_\xi .$$

The s is the arclength along the boundary. This can be substituted into the differentiated orthogonality condition to yield

$$s_\xi \left(\; \vec{z}_s \cdot \vec{r}_\eta + \vec{r}_s \cdot \vec{z}_\eta \; \right) = 0 \tag{6.24}$$

or

$$\vec{z}_s \cdot \vec{r}_\eta + \vec{r}_s \cdot \vec{z}_\eta = 0. \tag{6.25}$$

The derivative with respect to η can be approximated with one-sided differences

$$\vec{z}_s \cdot \left[\; 4 \; \vec{r}_{i,2} - 3 \; \vec{r}_{i,1} - \vec{r}_{i,3} \; \right] + \vec{r}_s \cdot \left[\; 4 \; \vec{z}_{i,2} - 3 \; \vec{z}_{i,1} - \vec{z}_{i,3} \; \right] = 0. \tag{6.26}$$

Along the boundary,

$$\vec{z}_{i,1} = \frac{\partial}{\partial \tau} \left[\; \vec{r}_{i,1}(s) \; \right] = \vec{r}_s \; s_\tau . \tag{6.27}$$

This can be substituted into the difference equation and solved for s_τ to yield

$$s_\tau = \frac{1}{3 \left(\vec{r}_s \cdot \vec{r}_s \right)} \left[\; \vec{z}_s \cdot \left(4 \; \vec{r}_{i,2} - 3 \; \vec{r}_{i,1} - \vec{r}_{i,3} \; \right) + \vec{r}_s \cdot \left(4 \; \vec{z}_{i,2} - \vec{z}_{i,3} \; \right) \; \right]. \tag{6.28}$$

Thus for each iteration, a new s_τ is computed and

$$s^{m+1} = s^m + \omega_{GSB} \; \Delta\tau \; s_\tau . \tag{6.29}$$

Thus, from the curve fit of the boundary,

$$\vec{r}^{\,n+1,m+1} = \vec{r}_{\text{fit}} \left(\; s^{m+1} \; \right). \tag{6.30}$$

Then the mesh speed at the boundary can be computed as

$$\vec{z}_B \; = \; \left(\; \vec{r}^{\,n+1,m+1} - \vec{r}^{\,n} \; \right) \; / \; \Delta\tau . \tag{6.31}$$

For the boundary along $j = NJ$,

$$s_\tau = \frac{1}{3\,(\vec{r}_s \cdot \vec{r}_s)} \quad [\quad \vec{z}_s \cdot \left(4\,\vec{r}_{i,NJ-1} - 3\,\vec{r}_{i,NJ} - \vec{r}_{i,NJ-2}\right) + \vec{r}_s \cdot \left(4\,\vec{z}_{i,NJ-1} - \vec{z}_{i,NJ-2}\right) \;]. \tag{6.32}$$

For the boundary along $i = 1$,

$$s_\tau = \frac{1}{3\,(\vec{r}_s \cdot \vec{r}_s)} \; \left[\; \vec{z}_s \cdot \left(4\,\vec{r}_{2,j} - 3\,\vec{r}_{1,j} - \vec{r}_{3,j}\right) + \vec{r}_s \cdot \left(4\,\vec{z}_{2,j} - \vec{z}_{3,j}\right) \; \right]. \tag{6.33}$$

For the boundary along $i = NI$,

$$s_\tau = \frac{1}{3\,(\vec{r}_s \cdot \vec{r}_s)} \quad [\quad \vec{z}_s \cdot \left(4\,\vec{r}_{NI-1,j} - 3\,\vec{r}_{NI,j} - \vec{r}_{NI-2,j}\right) + \vec{r}_s \cdot \left(4\,\vec{z}_{NI-1,j} - \vec{z}_{NI-2,j}\right) \;]. \tag{6.34}$$

Mixed boundary conditions are required when there is a prescribed boundary motion and the enforcement of orthogonality of mesh lines at the boundaries. The expression for the absolute velocity of the mesh point is determined from the particle dynamics. The velocity of a point P on a moving boundary is

$$\vec{V}_P = \vec{V}_B + \vec{V}_{BP} \tag{6.35}$$

where $\vec{V}_P$ is the absolute velocity, $\vec{V}_B$ is the velocity of the boundary at point B at the instant in time in which point P is coincident with point B, and $\vec{V}_{BP}$ is the velocity of point P relative to point B. Thus the mesh speed at the boundary is

$$\vec{z}_B(t) \; = \; \vec{z}_{B(motion)}(t) \; + \; \vec{z}_{BO}(t) \tag{6.36}$$

where $\vec{z}_{B(motion)}(t)$ is the prescribed motion of the boundary and $\vec{z}_{BO}(t)$ is the motion along the boundary due to requirements to maintain orthogonality of the mesh lines at the boundary.

Therefore, in computing s_τ for the orthogonality of the mesh lines at the boundary as presented in for the Neumann boundary conditions, the x_τ and y_τ mesh speeds used in the equations should now be mesh speeds relative to the boundary motion. The s^{m+1} and $\vec{r}^{\,n+1,m+1}$ are still computed and then

$$\vec{z}_B(t) \;=\; \vec{z}_{B(motion)}(t) \;+\; \left(\vec{r}^{\,n+1,m+1} - \vec{r}^{\,n} \right) \;/\; \Delta\tau. \tag{6.37}$$

The Mesh Control Law

Solving the mesh speed equations for the mesh speeds and then integrating to obtain the mesh is expected to generate a mesh which satisfies the mesh equations. However, small deviations may develop. One way to ensure that the mesh continues to satisfy the mesh equations is to rewrite the mesh speed equations in the form of a first-order homogeneous dynamic system [32] as

$$G_\tau \;+\; \lambda_C \, G \;=\; 0. \tag{6.38}$$

This is valid since we want G to be zero to satisfy the mesh equations. The λ can be regarded as a damping constant which damps deviations from the mesh equations. Equation 6.38 is solved using the Gauss-Seidel point relaxtion method in the form

$$\begin{aligned}
\vec{z}_{i,j}^{\,m+1} \;=\; & \frac{1}{2}\,(A+C)^{-1} \\
& \Big[\;\frac{1}{4}\,B\,(\vec{z}_{i+1,j+1}^{\,m} - \vec{r}_{i+1,j-1}^{\,m+1} - \vec{z}_{i-1,j+1}^{\,m} + \vec{r}_{i-1,j-1}^{\,m+1}) \\
& +\; A\,(\vec{z}_{i+1,j}^{\,m} + \vec{z}_{i-1,j}^{\,m+1}) + C\,(\vec{z}_{i,j+1}^{\,m} + \vec{z}_{i,j-1}^{\,m+1}) \\
& +\; \frac{1}{2}\,D\,(\vec{z}_{i+1,j}^{\,m} - \vec{z}_{i-1,j}^{\,m+1}) + \frac{1}{2}\,E\,(\vec{z}_{i,j+1}^{\,m} - \vec{z}_{i,j-1}^{\,m+1}) \\
& +\; \vec{T}^{\,m} + \lambda_C \, G \,].
\end{aligned} \tag{6.39}$$

CHAPTER 7. THE NUMERICAL PROCEDURES FOR SOLVING THE NAVIER-STOKES EQUATIONS

The semi-discrete, finite-volume form of the unsteady Navier-Stokes equations for the solution point (i,j) was presented in a previous chapter as equation (4.7),

$$\frac{d}{d\tau}\hat{U}(\tau)_{i,j} + \hat{R}(\tau)_{i,j} = 0. \tag{7.1}$$

This chapter will discuss the numerical methods for the time-integration of equation (7.1), the representation of the cell-face fluxes in $\hat{R}$, and the enforcement of the flow boundary conditions. The methods discussed here have been presented by various authors. This material is included not only for completeness, but also to show how these methods are modified when one allows the mesh to be dynamic.

The Time Discretization

The time integration of equation (7.1) is

$$\int_{\tau_1}^{\tau_2} \frac{d}{d\tau}\hat{U}(\tau)\, d\tau + \int_{\tau_1}^{\tau_2} \hat{R}(\tau)\, d\tau = 0. \tag{7.2}$$

The first term on the left-hand-side of equation (7.2) is an integration of an exact differential, so

$$\int_{\tau_1}^{\tau_2} \frac{d}{d\tau}\hat{U}(\tau)\, d\tau = \delta\hat{U}^n = \hat{U}^{n+1} - \hat{U}^n. \tag{7.3}$$

The superscript n denotes the τ_1 time level and the superscript $n+1$ denotes the τ_2 time level. The second term on the left-hand-side of equation (7.2) is approximated in the manner

$$\int_{\tau_1}^{\tau_2} \hat{R}(\tau)\, d\tau \;\approx\; \left[\, \theta\, \hat{R}^{\,n+1} + (1-\theta)\;\hat{R}^{\,n} \,\right]\, \Delta\tau \tag{7.4}$$

where $\Delta\tau = \tau_2 - \tau_1$. The θ allows for different types of integration approximations: for $\theta = 0$ the approximation is the Euler explicit method; for $\theta = 1/2$ the approximation is the trapezoidal method; for $\theta = 1$ the approximation is the Euler implicit method.

Thus the finite-volume form of the Navier-Stokes equations are now discretized as

$$\delta\hat{U}^n \;=\; -\,\Delta\tau \left[\, \theta\hat{R}^{\,n+1} + (1-\theta)\,\hat{R}^{\,n} \,\right]. \tag{7.5}$$

The Explicit, Lax-Wendroff Method

The Euler explicit time integration is formed from equation (7.5) if $\theta = 0$. This results in the finite-volume equation for cell (i,j) of the form

$$\hat{U}_{i,j}^{n+1} \;=\; \hat{U}_{i,j}^{n} \;-\; \Delta\tau\; \hat{R}_{i,j}^{n}. \tag{7.6}$$

The Euler explicit time integration is first-order in time. One method for obtaining higher-order accuracy in time is to use multiple stages to advance the solution from time level n to time level $n+1$.

A second-order accurate time integration can be achieved using the explicit, two-stage Lax-Wendroff method presented by Liou and Hsu [37],

$$\hat{U}_{i,j}^{*} \;=\; \hat{U}_{i,j}^{n} \;-\; \Delta\tau\; \hat{R}_{i,j}^{n}, \tag{7.7}$$

$$\hat{U}^{**}_{i,j} \;=\; \hat{U}^{*}_{i,j} \;-\; \Delta\tau\; \hat{R}^{*}_{i,j}, \tag{7.8}$$

and

$$\hat{U}^{n+1}_{i,j} = \frac{1}{2}\left[\, \hat{U}^{n}_{i,j} \;+\; \hat{U}^{**}_{i,j} \,\right]. \tag{7.9}$$

The Mathematical Character of the Navier-Stokes and Euler Equations

The details of the use of the explicit equations (7.7-7.9) in computing the solution of the Navier-Stokes and Euler equations require understanding the mathematical character of the Navier-Stokes and Euler equations. This is discussed in this section.

The mathematical character of the Navier-Stokes and Euler equations is examined by looking at the quasilinear form of equation (7.1),

$$\frac{d}{d\tau}\,\delta\hat{U}(\tau)_{\,i,j} \;+\; \frac{\partial \hat{R}}{\partial \hat{U}}\,\delta\hat{U}(\tau)_{\,i,j} \;=\; 0. \tag{7.10}$$

This equation is a second-order equation for the Navier-Stokes equations and a first-order equation for the Euler equations. The $\frac{\partial \hat{R}}{\partial \hat{U}}$ is the Jacobian matrix of the flux vectors for the finite-volume cell. Thus

$$\left.\frac{\partial \hat{R}}{\partial \hat{U}}\right|_{i,j} \delta\hat{U}_{\,i,j} \;=\; \left(\hat{A}\delta\hat{U}\right)_{i+1/2,j} \;-\; \left(\hat{A}\delta\hat{U}\right)_{i-1/2,j} \tag{7.11}$$

$$+\; \left(\hat{B}\delta\hat{U}\right)_{i,j+1/2} \;-\; \left(\hat{B}\delta\hat{U}\right)_{i,j-1/2}. \tag{7.12}$$

The Jacobians on the cell faces are defined in the manner of

$$\hat{A}_{i+1/2,j} \;=\; \frac{\partial \hat{F}_{i+1/2,j}}{\partial \hat{U}} \;=\; \left[\, (\,\vec{A}\cdot\hat{n}dS\,)\,/\,V\,\right]_{i+1/2,j}. \tag{7.13}$$

The Jacobian $\hat{A}_{i-1/2,j}$ is computed in a similar manner. The $\hat{B}$ Jacobians are on the cell faces in the η direction. The Jacobian $\hat{B}_{i,j+1/2}$ is computed in the manner of

$$\hat{B}_{i,j+1/2} = \frac{\partial \hat{F}_{i,j+1/2}}{\partial \hat{U}} \;=\; \left[\, (\,\vec{A}\cdot\hat{n}dS\,)\,/\,V\,\right]_{i,j+1/2}. \tag{7.14}$$

The Jacobian $\hat{B}_{i,j-1/2}$ is computed in a similar manner.

The $\vec{A}$ is the vector of Jacobians of the Cartesian fluxes

$$\vec{A} = (A - A_V) \hat{i} + (B - B_V) \hat{j} \tag{7.15}$$

where

$$A = \frac{\partial E}{\partial U}, \quad A_V = \frac{\partial E_V}{\partial U},$$
$$B = \frac{\partial F}{\partial U}, \quad \text{and} \quad B_V = \frac{\partial F_V}{\partial U}.$$

The numerical methods to be presented in the following sections are primarily based on the wave nature of the inviscid flux terms. Therefore, it is assumed for the following discussions that there are no viscous terms in $\hat{A}$ and $\hat{B}$. For conciseness, consider only the fluxes in the ξ direction and examine the mathematical character of $\hat{A}$. The analysis of $\hat{B}$ is similar and will not be shown.

The eigenvalues of $\hat{A}$ are determined from the equation

$$\det \left| \hat{\lambda}_m I - \hat{A} \right| = 0 \tag{7.16}$$

and are of the form

$$\hat{\lambda}_m = \frac{S}{V} \lambda_m, \tag{7.17}$$

where

$$\lambda_1 = V_{RN} = (\vec{V} - \vec{g}) \cdot \hat{n}, \tag{7.18}$$

$$\lambda_2 = V_{RN} = (\vec{V} - \vec{g}) \cdot \hat{n}, \tag{7.19}$$

$$\lambda_3 = V_{RN} + c = (\vec{V} - \vec{g}) \cdot \hat{n} + c, \tag{7.20}$$

and

$$\lambda_4 = V_{RN} - c = (\vec{V} - \vec{g}) \cdot \hat{n} - c. \tag{7.21}$$

Note that all eigenvalues are real numbers, and thus, the unsteady, Euler equations are hyperbolic equations. The λ_1 and λ_2 are the convective wave speeds, while λ_3 and λ_4 are the acoustic wave speeds.

A similarity transformation exists for $\hat{A}$ of the form

$$\hat{A} \;=\; P\,\Lambda\,P^{-1}. \tag{7.22}$$

The Λ is a diagonal matrix of the eigenvalues. The columns of P are the right eigenvectors $\hat{\vec{r}}_m$ associated with each eigenvalue λ_m of $\hat{A}$. A wave moves in a direction tangent to the right eigenvector at a wave speed of λ_m. The matrix P is

$$P = \begin{bmatrix} 1 & 0 & \rho/2c & \rho/2c \\ u & n_2\rho & \rho(u+cn_1)/2c & \rho(u-cn_1)/2c \\ v & -n_1\rho 0 & \rho(v+cn_2)/2c & \rho(v-cn_2)/2c \\ V^2/2 & \rho(un_2-vn_1) & \rho(h_t+c\vec{V}\cdot\hat{n})/2c & \rho(h_t-c\vec{V}\cdot\hat{n})/2c \end{bmatrix}. \tag{7.23}$$

The rows of P^{-1} are the left eigenvectors $\hat{\vec{l}}_m$ associated with each eigenvalue of $\hat{A}$. The matrix P^{-1} is

$$P^{-1} = \begin{bmatrix} 1-\frac{\gamma-1}{2}M^2 & (\gamma-1)u/c^2 & (\gamma-1)v/c^2 & -(\gamma-1)/c^2 \\ (vn_1-un_2)/\rho & n_2/\rho & -n_1/\rho & 0 \\ \Upsilon^- & \Omega_1^- & \Omega_2^- & \frac{\gamma-1}{\rho c} \\ \Upsilon^+ & \Omega_1^+ & \Omega_2^+ & \frac{\gamma-1}{\rho c} \end{bmatrix} \tag{7.24}$$

where

$$\Upsilon^{\pm} \;=\; c\left(\frac{\gamma-1}{2}\,M^2 \;\pm\; \vec{V}\cdot\hat{n}\;/\;c\right)\;/\;\rho, \tag{7.25}$$

$$\Omega_1^{\pm} \;=\; \mp\left[\,n_1 \;\pm\; (\gamma-1)\,\frac{u}{c}\,\right]\;/\;\rho, \tag{7.26}$$

and

$$\Omega_2^{\pm} = \mp \left[n_2 \pm (\gamma - 1) \frac{v}{c} \right] / \rho. \tag{7.27}$$

The unit normal to the cell face is

$$\hat{n} = n_1 \, \hat{i} + n_2 \, \hat{j}. \tag{7.28}$$

These eigenvectors have been scaled such that they are orthonormal.

Using the previous information, the inviscid cell-face flux can be written as

$$\hat{F} = \hat{A} \, \hat{U} = P \, \Lambda \, P^{-1} \, \hat{U} = \sum_{m=1}^{4} \lambda_m \, \hat{r}_m \, \hat{l}_m^T \, \hat{U}. \tag{7.29}$$

In the above analysis, the effects of the mesh speeds are present only in the eigenvalues. This will allow for a straightforward application of the Roe flux-difference splitting method, which will be discussed below.

The propagation of information in the solution is understood further through the characteristic formulation of the Euler equations. Compatibility relations can be formulated which relate changes in the solution along the characteristic lines. These compatibility relations are derived by premultiplying equation (7.10) by the matrix of left eigenvectors, P^{-1}. This results in the compatibility relations in terms of the differentials of the characteristic variables,

$$\frac{d}{d\tau} (\, \delta \hat{W}(\tau) \,)_{i,j} + \frac{\partial \hat{R}}{\partial \hat{U}} (\, \delta \hat{W}(\tau) \,)_{i,j} = 0 \tag{7.30}$$

where the differentials of the characteristic variables are defined by

$$\delta \hat{W} = P^{-1} \, \delta \hat{U}. \tag{7.31}$$

The algebraic vector of the differentials of the characteristic variables is

$$\delta \hat{W}^T = V \, [\, \delta w_1, \, \delta w_2, \, \delta w_3, \, \delta w_4 \,] \tag{7.32}$$

where, the δw_m are defined by

$$\delta w_m = \tilde{l}_m^T \Delta U \tag{7.33}$$

and are a measure of the strength of the wave. For the two-dimensional Euler equations, the δw_m expressions are

$$\delta w_1 = \Delta\rho - \Delta p / c^2, \tag{7.34}$$

$$\delta w_2 = \hat{n}_y \Delta u - \hat{n}_x \Delta v, \tag{7.35}$$

$$\delta w_3 = \hat{n}_x \Delta u + \hat{n}_y \Delta v + \Delta p / \rho c, \tag{7.36}$$

and

$$\delta w_4 = -(\hat{n}_x \Delta u + \hat{n}_y \Delta v) + \Delta p / \rho c. \tag{7.37}$$

This information will be used below in the formulation of the Roe flux-difference splitting method and in the boundary conditions.

The CFL Condition

The stability properties of the two-stage Lax-Wendroff method indicate that the time step used for the time-marching is limited by the CFL condition.

The CFL condition [34] for multi-dimensional problems using generalized coordinates can be stated as

$$\Delta\tau = \min\{ \Delta\tau_\xi, \Delta\tau_\eta \} \tag{7.38}$$

where

$$\Delta\tau_\xi = \frac{\nu \Delta\xi}{| \hat{\lambda}_\xi |max}. \tag{7.39}$$

The subscript max denotes the maximum absolute value calculated among all the solution points. The ν is the CFL number. The $\hat{\lambda}_\xi$ are the eigenvalues along the ξ-coordinate direction. There are similar expressions for the η-coordinate direction.

The Explicit, Inviscid Flux Formula

The time integration methods presented above require some method for determining the numerical fluxes, $\hat{F}$, in $\hat{R}_{i,j}$. This is a multi-dimensional problem. However, the common approach is to compute the flux as a one-dimensional problem along the coordinate lines, which are approximately normal to the cell face.

The obvious approach for computing $\hat{F}$ would be to simply average the solution values from each side of the face along the coordinate line. However, this approach is unstable unless some form of dissipation is added to the flux. This is done through the choice of how the solution information about the cell face is used in the numerical flux formula. Central schemes obtain the information in a manner symmetric to the cell face. Upwind schemes use the wave propagation properties at the cell face to formulate the numerical flux. Upwind refers to the idea that information for computing the flux comes from directions along the characteristics. One popular upwind flux method has been Roe's flux-difference splitting method [40]. This method computes the fluxes by approximating the solution of the Riemann problem at each cell face.

The Roe's flux-difference splitting method begins with the splitting

$$\hat{F}_{i+1/2,j} = \hat{F}^+_{i+1/2,j} + \hat{F}^-_{i+1/2,j} = \hat{F}^+_{i,j} + \hat{F}^-_{i+1,j} \qquad (7.40)$$

or

$$\hat{F}_{i+1/2,j} = \frac{1}{2}\left(\hat{F}_{i,j} + \hat{F}_{i+1,j}\right) + \frac{1}{2}\Delta\hat{F}^{-}_{i+1/2,j} - \frac{1}{2}\Delta\hat{F}^{+}_{i+1/2,j}. \tag{7.41}$$

The flux differences are defined by a decomposition of the waves

$$\Delta\hat{F} = \hat{F}_R - \hat{F}_L = \sum_{m=1}^{4} \lambda_m \hat{\vec{r}}_m \delta w_m. \tag{7.42}$$

Further,

$$\Delta\hat{F} = \Delta\hat{F}^{+} + \Delta\hat{F}^{-}, \tag{7.43}$$

where

$$\Delta\hat{F}^{+} = \sum_{m}^{4} \lambda_m^{(+)} \hat{\vec{r}}_m \delta w_m \tag{7.44}$$

and

$$\Delta\hat{F}^{-} = \sum_{m}^{4} \lambda_m^{(-)} \hat{\vec{r}}_m \delta w_m. \tag{7.45}$$

For the computation of the δw_m, the differentials of the primitive variables across the cell face are defined by

$$\Delta p = p_R - p_L, \tag{7.46}$$

$$\Delta\rho = \rho_R - \rho_L, \tag{7.47}$$

$$\Delta u = u_R - u_L, \tag{7.48}$$

and

$$\Delta v = v_R - v_L. \tag{7.49}$$

The subscript L denotes the state at the left or backward side of the cell face with the forward direction being along the coordinate line. The subscript R denotes the right or forward side of the cell face.

The values of the flow variables used in computing the eigenvalues and eigenvectors at the cell face are computed in a manner such that they maintain what Roe calls property U. This requires that the Jacobian matrix should be computed exactly in the case of a uniform solution across the cell face, that the flow conservation be maintained, and that the Rankine-Hugoniot jump relations are satisfied. This leads to the definition of Roe-averaged properties across the cell face,

$$\rho^2 = \rho_L \, \rho_R, \tag{7.50}$$

$$u = \frac{\rho_L^{1/2} \, u_L + \rho_R^{1/2} \, u_R}{\rho_L^{1/2} + \rho_R^{1/2}}, \tag{7.51}$$

$$v = \frac{\rho_L^{1/2} \, v_L + \rho_R^{1/2} \, v_R}{\rho_L^{1/2} + \rho_R^{1/2}}, \tag{7.52}$$

$$h_t = \frac{\rho_L^{1/2} \, (h_t)_L + \rho_R^{1/2} \, (h_t)_R}{\rho_L^{1/2} + \rho_R^{1/2}}, \tag{7.53}$$

and

$$c^2 = (\gamma - 1) \left[h_t - \frac{1}{2} \left(u^2 + v^2 \right) \right]. \tag{7.54}$$

Roe's method approximates the solution of the one-dimensional Euler equations across the cell face. It is known that expansion shocks are a valid solution to the Euler equations, but are not a physically possible solution since the second law of thermodynamics would be violated. Thus Roe's method has the disadvantage that expansion shocks may be computed. One correction is to modify the eigenvalues near sonic conditions. The modification is of the form

$$| \lambda | = \frac{\lambda^2 + \epsilon^2}{2 \, \epsilon}, \tag{7.55}$$

when

$$| \lambda | < \epsilon, \tag{7.56}$$

where

$$\epsilon = K \max (\lambda_R - \lambda_L, 0), \tag{7.57}$$

and

$$K > 0.$$

Equation 7.41 produces a first-order flux. A second-order flux formula can be obtained through a linear extrapolation of the flux differences from the upwind directions to the cell face. However, such an extrapolation results in oscillations about discontinuities in the solution. These may result in the computation of nonphysical properties and at the least, they degrade the resolution of the solution. One approach for removing the oscillations is to limit the change in the flux difference if the flux differences vary along the coordinate line. This approach is known as Total Variational Diminishing (TVD) limiting. Thus, the formula for the $(i + 1/2, j)$ face is

$$\begin{aligned} \hat{F}_{i+1/2,j} &= \frac{1}{2}\left\{\hat{F}_{i,j} + \hat{F}_{i+1,j}\right\} - \frac{1}{2}\Delta\hat{F}^{+}_{i+1/2,j} + \frac{1}{2}\Delta\hat{F}^{-}_{i+1/2,j} \\ &+ \frac{1}{2}\left\{\left(\frac{\Delta s_i}{\Delta s_{i-1}}\right)\psi\left(r^{+}_{i-1/2}\right)\Delta\hat{F}^{+}_{i-1/2,j}\right\} \\ &- \frac{1}{2}\left\{\left(\frac{\Delta s_i}{\Delta s_{i+1}}\right)\psi\left(r^{+}_{i+3/2}\right)\Delta\hat{F}^{-}_{i+3/2,j}\right\}. \end{aligned} \tag{7.58}$$

There are similar formulas for the flux on the other faces. Equation 7.58 accounts for the non-uniformity of the mesh along a coordinate line. The change in the arclength along the coordinate line is

$$\Delta s_i = s_{i+1,j} - s_{i,j} = \left\{\left(x_{i+1,j} - x_{i,j}\right)^2 + \left(y_{i+1,j} - y_{i,j}\right)^2\right\}^{1/2}. \tag{7.59}$$

The $\psi(r)$ is the TVD limiter. One popular limiter is Roe's Superbee limiter,

$$\psi(r) = \max[\ 0,\ \min(2r, 1),\ \min(r, 2)\ ]. \tag{7.60}$$

Another limiter is an exponential limiter

$$\psi(r) = \max\left\{ \frac{2^n r}{(1 + |r|^{1/n})^n},\ 0 \right\}. \tag{7.61}$$

For n=1, the limiter is the van Leer harmonic limiter. For n=2, the limiter is the Roe Hyperbee limiter.

The ratios of flux differences are defined for the system of equations as

$$r^+_{i-1/2} = \frac{\langle\ \Delta\hat{F}^+_{i+1/2},\ \Delta\hat{F}^+_{i-1/2}\rangle}{\langle\ \Delta\hat{F}^+_{i-1/2},\ \Delta\hat{F}^+_{i-1/2}\rangle} \tag{7.62}$$

and

$$r^-_{i+3/2} = \frac{\langle\ \Delta\hat{F}^-_{i+3/2},\ \Delta\hat{F}^-_{i+1/2}\rangle}{\langle\ \Delta\hat{F}^-_{i+3/2},\ \Delta\hat{F}^-_{i+3/2}\rangle}. \tag{7.63}$$

The $\langle\ \rangle$ denotes a scalar product of the flux vectors.

The Explicit, Viscous Flux Formula

This section discusses the numerical procedures for computing the viscous component of the cell-face flux, $\hat{F}_V$, where

$$\hat{F}_V\ =\ \hat{n} \cdot \left(\ E_V\ \hat{\imath}\ +\ F_V\ \hat{\jmath}\ \right). \tag{7.64}$$

This involves approximating partials of u, v, and T with respect to x and y at the cell face. The chain rule shows that

$$\frac{\partial(\ )}{\partial x} = \left(y_\eta\ \frac{\partial(\ )}{\partial \xi} - y_\xi\ \frac{\partial(\ )}{\partial \eta} \right)\ /\ J \tag{7.65}$$

and

$$\frac{\partial(\ )}{\partial y} = \left(x_\xi \frac{\partial(\ )}{\partial \eta} - x_\eta \frac{\partial(\ )}{\partial \xi} \right) \ / \ J. \tag{7.66}$$

On the $i+1/2$ face, the derivatives with respect to ξ are computed in a straightforward manner with a central difference of the form

$$x_\xi \ \approx \ x_{i+1,j} \ - \ x_{i,j}, \tag{7.67}$$

which is second-order with respect to the cell-face. Similar expressions exist for y_ξ, u_ξ, v_ξ, and T_ξ, The derivatives with respect to η are computed as averages of the derivatives with respect to η at i and $i+1$,

$$x_\eta \ \approx \ \frac{1}{2} \ \left[\ (x_\eta)_{i,j} \ + \ (x_\eta)_{i+1,j} \ \right] \tag{7.68}$$

or

$$x_\eta \ \approx \ \frac{1}{4} \ \left(\ x_{i,j+1} \ - \ x_{i,j-1} \ + \ x_{i+1,j+1} \ - \ x_{i+1,j-1} \ \right). \tag{7.69}$$

Similar expressions exist for y_η, u_η, v_η and T_η. On the $j+1/2$ face, the formulas are analogous as for the $i+1/2$ face. The values of u, v, μ_l, and μ_t at the cell face are computed using an arithmetic average of the values on either side of the cell face along the coordinate line.

Flow Boundary Conditions

The two-dimensional flow domain consists of four boundaries. Boundaries (1) and (3) are constant-ξ boundaries with $i = 1$ and $i = NI$, respectively. Boundaries (2) and (4) are constant-η boundaries with $j = NJ$ and $j = 1$, respectively. This section discusses the numerical procedures for imposing the flow conditions at the boundaries for several types of flowfields. The procedures follow the common approach of using

characteristic information; however, for this work, the procedures are extended for the case of a dynamic mesh.

Two-dimensional boundary geometry

The unit normal vector at the boundary, $\hat{n}_B$,

$$\hat{n}_B = n_1\ \hat{\imath} + n_2\ \hat{\jmath} \tag{7.70}$$

is considered to be positive as pointing into the flow domain.

The unit tangent vector at the boundary, $\hat{t}_B$,

$$\hat{t}_B = t_1\ \hat{\imath} + t_2\ \hat{\jmath} \tag{7.71}$$

is considered to be directed in the positive ξ or η directions.

The unit normal and tangential vectors are related through the condition

$$\hat{t} \cdot \hat{n} = 0. \tag{7.72}$$

Along boundaries (1) and (2) another condition is

$$\hat{n} \times \hat{t} = \hat{k}, \tag{7.73}$$

while along boundaries (3) and (4) the condition is

$$\hat{t} \times \hat{n} = \hat{k}. \tag{7.74}$$

Thus along boundaries (1) and (2) it can be shown that

$$t_1 = -\ n_2 \quad \text{and} \quad t_2 = n_1, \tag{7.75}$$

while along boundaries (3) and (4) it can be shown that

$$t_1 = n_2 \quad \text{and} \quad t_2 = -\ n_1. \tag{7.76}$$

At boundary (1),

$$\hat{n}_{B1} = \frac{\nabla\xi}{|\nabla\xi|} = \frac{\xi_1\,\hat{i} + \xi_2\,\hat{j}}{\left(\xi_1^2 + \xi_2^2\right)^{1/2}} \tag{7.77}$$

where

$$\xi_1 = n_1 = \frac{1}{2}\left(y_{1,j+1} - y_{1,j-1}\right) \tag{7.78}$$

and

$$\xi_2 = n_2 = -\frac{1}{2}\left(x_{1,j+1} - x_{1,j-1}\right). \tag{7.79}$$

At boundary (2),

$$\hat{n}_{B2} = -\frac{\nabla\eta}{|\nabla\eta|} = -\frac{\eta_1\,\hat{i} + \eta_2\,\hat{j}}{\left(\eta_1^2 + \eta_2^2\right)^{1/2}} \tag{7.80}$$

where

$$\eta_1 = n_1 = -\frac{1}{2}\left(y_{i+1,NJ} - y_{i-1,NJ}\right) \tag{7.81}$$

and

$$\eta_2 = n_2 = \frac{1}{2}\left(x_{i+1,NJ} - x_{i-1,NJ}\right). \tag{7.82}$$

At boundary (3),

$$\hat{n}_{B3} = -\frac{\nabla\xi}{|\nabla\xi|} = -\frac{\xi_1\,\hat{i} + \xi_2\,\hat{j}}{\left(\xi_1^2 + \xi_2^2\right)^{1/2}} \tag{7.83}$$

where

$$\xi_1 = n_1 = \frac{1}{2}\left(y_{NI,j+1} - y_{NI,j-1}\right) \tag{7.84}$$

and

$$\xi_2 = n_2 = -\frac{1}{2}\left(x_{NI,j+1} - x_{NI,j-1}\right). \tag{7.85}$$

At boundary (4),

$$\hat{n}_{B4} = \frac{\nabla\eta}{|\nabla\eta|} = \frac{\eta_1\,\hat{i} + \eta_2\,\hat{j}}{\left(\eta_1^2 + \eta_2^2\right)^{1/2}} \tag{7.86}$$

where

$$\eta_1 = n_1 = -\frac{1}{2}\left(y_{i+1,1} - y_{i-1,1}\right) \tag{7.87}$$

and

$$\eta_2 = n_2 = \frac{1}{2}\left(x_{i+1,1} - x_{i-1,1}\right). \tag{7.88}$$

The normal velocity at the boundary is

$$V_{NB} = \vec{V}_B \cdot \hat{n}_B = u\, n_1 + v\, n_2. \tag{7.89}$$

A positive normal velocity at the boundary indicates an inflow point on the boundary, while a negative normal velocity indicates an outflow point. The tangential velocity at the boundary is

$$V_{TB} = \vec{V}_B \cdot \hat{t}_B = u\, t_1 + v\, t_2. \tag{7.90}$$

One can show that

$$u = n_1\, V_N + t_1\, V_T \tag{7.91}$$

and

$$v = n_2\, V_N + t_2\, V_T. \tag{7.92}$$

The enforcement of boundary conditions often requires an extrapolation of data from the interior to the boundary. This section discusses the method of extrapolation for nonuniform meshes. The mesh is assumed to be a cell-vertex formulation with solution points at the boundaries. The first-order extrapolation from the left to the boundary is

$$\left(\phi_B\right)_{extrap} = \left(1 + \alpha_L\right)\phi_{B-1} - \alpha_L\, \phi_{B-2} \tag{7.93}$$

where

$$\alpha_L = \frac{\Delta s_{B-1}}{\Delta s_{B-2}} \tag{7.94}$$

and ϕ is the extrapolated quantity. The extrapolation from the right to the boundary is

$$\left(\phi_B\right)_{extrap} = \left(1 + \alpha_R\right) \phi_{B+1} - \alpha_R \, \phi_{B+2} \tag{7.95}$$

where

$$\alpha_R = \frac{\Delta s_B}{\Delta s_{B+1}}. \tag{7.96}$$

The α coefficients account for non-uniformities along the direction of extrapolation. For an equally spaced mesh,

$$\alpha_L = \alpha_R = 1. \tag{7.97}$$

For generalized coordinates, the extrapolations are usually performed along constant-ξ or constant-η coordinate lines. The Δs are arc lengths along the coordinate lines. The Δs_B length is defined as

$$\Delta s_B = \mid \vec{r}_{B+1} - \vec{r}_B \mid = \; s_{B+1} - s_B. \tag{7.98}$$

For a zeroth-order extrapolation,

$$\alpha_L \; = \; \alpha_R \; = \; 0. \tag{7.99}$$

Characteristic boundary conditions

The characteristic boundary condition method considers the incoming and outgoing characteristics normal to the boundary. A positive eigenvalue (wave speed) indicates a wave entering the flow domain, and so a physical boundary condition must be specified. A negative eigenvalue indicates a wave leaving the flow domain, and so a numerical boundary condition must be specified. The waves moving tangential to the boundary are neglected in the boundary condition treatment.

The compatibility relations relate the changes in properties along a characteristic. The compatibility relations for an arbitrary direction are

$$d_b^{\pm} R_n^{\pm} = \mp c\, \hat{t} \cdot \left(\hat{t} \cdot \nabla \right) \vec{V}, \tag{7.100}$$

$$d_b^{\pm} = \frac{\partial}{\partial t} + \left(\vec{V} \pm c\, \hat{n} \right) \cdot \vec{\nabla}, \tag{7.101}$$

and

$$R_n^{\pm} = (\vec{V} - \vec{g}) \cdot \hat{n} \pm \frac{2\, c}{\gamma - 1}. \tag{7.102}$$

For a stationary boundary on which the flow does not vary along the boundary, the left hand side of the compatiblity relations for the acoustic waves becomes zero and $R_n^{\pm}$ becomes an invariant, which is known as a Riemann invariant,

$$R_n^{\pm} = V_{RNB} \pm \frac{2\, c}{\gamma - 1}. \tag{7.103}$$

The subsonic inflow boundary

For an inflow boundary,

$$V_{RNB} > 0. \tag{7.104}$$

Further, for subsonic inflow,

$$\lambda_1, \lambda_2, \lambda_3 > 0$$

and

$$\lambda_4 < 0.$$

Thus three physical and one numerical boundary conditions must be imposed. The flow along the boundary is assumed to be uniform and so the Riemann invariants can be computed.

The numerical boundary condition is imposed through the R^- characteristic value evaluated from the interior flow domain,

$$R^-_{RNB} = V_{RN(extrap)} - \frac{2\, c_{(extrap)}}{\gamma - 1}. \tag{7.105}$$

The subscript $(extrap)$ denotes that the value is extrapolated from the interior flow domain.

The first physical boundary condition imposed through the R^+ characteristic value evaluated using the physical state of the inflow;

$$R^+_{RNB} = V_{RN(inflow)} + \frac{2\, c_{(inflow)}}{\gamma - 1}, \tag{7.106}$$

where the subscript $(inflow)$ denotes that the value is obtained from the inflow state.

Note that at the boundary,

$$R^+_{RNB} = V_{RNB} + \frac{2\, c_B}{\gamma - 1} \tag{7.107}$$

and

$$R^-_{RNB} = V_{RNB} - \frac{2\, c_B}{\gamma - 1}. \tag{7.108}$$

Thus,

$$V_{RNB} = \frac{1}{2} \left(R^+_{RNB} + R^-_{RNB} \right) \tag{7.109}$$

and

$$c_B = \frac{\gamma - 1}{4} \left(R^+_{RNB} - R^-_{RNB} \right). \tag{7.110}$$

The second physical boundary condition is imposed by evaluating the tangential velocity at the boundary from the inflow conditions

$$V_{RTB} = V_{RT(inflow)} = t_1 \left(u_{(inflow)} - x_\tau \right) + t_2 \left(v_{(inflow)} - y_\tau \right). \tag{7.111}$$

The third physical boundary condition is imposed by setting the entropy at the boundary equal to the value from the inflow state. Thus,

$$s_B = s_{(inflow)}. \tag{7.112}$$

This condition is equivalent to propagating the characteristic value associated with the first eigenvalue, since the compatibility relation associated with the first eigenvalue represents the convection of entropy.

Thus, knowing V_{NB}, V_{TB}, c_B, and s_B, the solution at the boundary can be computed

$$u_B = n_1 V_{RNB} + t_1 V_{RTB} + (x_\tau)_B, \tag{7.113}$$

$$v_B = n_2 V_{RNB} + t_2 V_{RTB} + (y_\tau)_B, \tag{7.114}$$

$$e_B = c_B^2 / \gamma (\gamma - 1), \tag{7.115}$$

and

$$\rho_B = \left[\frac{e_B (\gamma - 1)}{s_B} \right]^{\frac{1}{\gamma - 1}}. \tag{7.116}$$

An alternative to computing c_B from the characteristic values is to compute c_B from the inflow total enthalpy,

$$c_B^2 = (\gamma - 1) \left[h_{t(inflow)} - \frac{1}{2} V_B^2 \right], \tag{7.117}$$

where

$$V_B^2 = u_B^2 + v_B^2. \tag{7.118}$$

The *inflow* conditions are known according to type of flow problem. For external flow problems, the freestream conditions are known and used as the inflow conditions. For internal flow problems, the flow conditions are usually known at the inflow plane of the domain and these are used as the inflow conditions.

For the internal flow problems considered in this work, the total pressure and total temperature are specified at the inlet. These are used as two physical boundary conditions. A third physical boundary condition is established by requiring that the v component of velocity be zero. To be consistent with characteristic theory, the R^- invariant is computed from the interior data. Thus the boundary states can be computed. This requires solving a nonlinear equation for the temperature at the boundary. A Newton iteration is used.

The supersonic inflow boundary

For the supersonic inflow boundary,

$$V_{RNB} > 0 \tag{7.119}$$

and

$$\lambda_1,\ \lambda_2,\ \lambda_3,\ \lambda_4 > 0.$$

Thus, all boundary conditions are physical and the boundary solution can be prescribed from the inflow state.

The subsonic outflow boundary

For an outflow boundary

$$V_{RNB} < 0. \tag{7.120}$$

Further, for the subsonic outflow boundary

$$\lambda_1,\ \lambda_2,\ \lambda_4 < 0$$

and

$$\lambda_3 > 0.$$

Thus, one physical boundary condition must be imposed from the outflow state. The other boundary conditions must be imposed from the interior flow domain.

The first numerical boundary condition is imposed in the form

$$R^{-}_{RNB} = V_{RN(extrap)} - \frac{2\,c_{(extrap)}}{\gamma - 1}. \tag{7.121}$$

The R^{-} characteristic value must be obtained from the interior flow domain. The subscript $(extrap)$ denotes that the value is extrapolated from the interior flow domain.

The second numerical boundary condition can be imposed through the extrapolation of the tangential velocity from the interior flow domain. Thus

$$V_{RTB} = V_{RT(extrap)}. \tag{7.122}$$

The third numerical boundary condition can be imposed by extrapolating the entropy from the interior flow domain. Thus,

$$s_B = s_{(extrap)}. \tag{7.123}$$

The physical boundary condition can be imposed by evaluating the R^{+} characteristic value from the physical state of the outflow,

$$R^{+}_{RNB} = V_{RN(outflow)} + \frac{2\,c_{(outflow)}}{\gamma - 1} \tag{7.124}$$

where the subscript $(outflow)$ denotes the outflow state. Again the use of the invariants is based on the assumption that the outflow state is uniform in the boundary surface.

Then, at the boundary,

$$R^{+}_{RNB} = V_{RNB} + \frac{2\,c_B}{\gamma - 1} \tag{7.125}$$

and

$$R^{-}_{RNB} = V_{RNB} - \frac{2\,c_B}{\gamma - 1}. \tag{7.126}$$

Thus,

$$V_{RNB} = \frac{1}{2}\left(R^{+}_{RNB} + R^{-}_{RNB} \right) \tag{7.127}$$

and

$$c_B = \frac{\gamma - 1}{4}\left(R^{+}_{RNB} - R^{-}_{RNB} \right). \tag{7.128}$$

Thus, knowing V_{RNB}, V_{RTB}, c_B, and s_B, the solution at the boundary can be computed

$$u_B = n_1\, V_{RNB} + n_2\, V_{RTB} + (x_\tau)_B, \tag{7.129}$$

$$v_B = t_1\, V_{RNB} + t_2\, V_{RTB} + (y_\tau)_B, \tag{7.130}$$

$$e_B = c_B^2 \,/\, \gamma\,(\gamma - 1), \tag{7.131}$$

and

$$\rho_B = \left[\frac{e_B\,(\gamma - 1)}{s_B} \right]^{1/(\gamma - 1)}. \tag{7.132}$$

Another way of imposing the physical boundary condition is to prescribe a static pressure at the outflow, $p_{(outflow)}$,

$$p_B = p_{outflow}. \tag{7.133}$$

Thus

$$\rho_B = \left(p_B \,/\, s_B \right)^{1/\gamma}, \tag{7.134}$$

$$c_B = \left(\gamma\, p_B \,/\, \rho_B \right)^{1/2}, \tag{7.135}$$

$$V_{RNB} = R^{-}_{RNB} + 2\,c_B \,/\, (\gamma - 1), \tag{7.136}$$

$$u_B = n_1\, V_{RNB} + n_2\, V_{RTB} + (x_\tau)_B, \tag{7.137}$$

$$v_B = t_1 V_{RNB} + t_2 V_{RTB} + (y_\tau)_B, \tag{7.138}$$

and

$$e_B = p_B / \rho_B (\gamma - 1). \tag{7.139}$$

Thus ρ_B, e_B, u_B, and v_B are known and the solution is known.

The $p_{(outflow)}$ can be a steady value or vary with time. As long as the pressure does not vary along the outflow boundary, one can use the relations above. The forms of the time-varying pressures used in this work include an impulse, a step, and a sinusoidal variation.

It may not be valid to specify the outflow pressure in the case when vortices interact with the outflow boundary. Also, imposing the pressure may cause reflection of outgoing waves back into the flow domain. Nonreflecting boundary conditions may be needed to damp the exit pressure to the prescribed value.

The supersonic outflow boundary

For the supersonic outflow boundary,

$$V_{RNB} < 0 \tag{7.140}$$

and

$$\lambda_1, \lambda_2, \lambda_3, \lambda_4 < 0.$$

Thus all boundary conditions are numerical and must be determined from the interior flow domain. One method is to simply extrapolate the conserved variables, U. The extrapolation formulas discussed above can be used directly with $\phi = U$. However, it is felt that extrapolating the primitive variables or characteristic variables works as well. Computational efficiency will probably dictate which is the best approach.

The extrapolations are not mathematically correct across shock waves and other discontinuities. However, in practice the extrapolations are made anyway without introducing significant error.

The solid, inviscid wall boundary condition

The physical boundary condition for a solid, inviscid wall is

$$\rho_B \left(\vec{V}_B - \vec{g}_B \right) \cdot \hat{n} = \rho_B V_{RNB} = \dot{m}_B. \tag{7.141}$$

where $\vec{g}_B$ is the velocity of the boundary. This states that the mass flow rate at the wall is specified. If there is mass flow (i.e., suction or blowing), then the relative normal velocity can be positive or negative and the boundary becomes an inflow or outflow boundary and the methods of the previous section can be used.

If there is no mass flow, then the physical boundary condition reduces to

$$V_{RNB} = 0 \tag{7.142}$$

and

$$\lambda_1 = \lambda_2 = 0,$$

and

$$\lambda_3 > 0 \quad \text{and} \quad \lambda_4 < 0.$$

Thus, the convective eigenvalues are zero and there is one positive and one negative acoustic eigenvalue. This results in three numerical boundary conditions and one physical boundary condition.

Note that for the discrete case, $\vec{g}_B$ may be nonzero even for a stationary boundary; the mesh points on the boundary may be in motion as part of the mesh movement procedure.

The first numerical boundary condition is imposed by extrapolating the relative tangential velocity,

$$V_{RTB} = (V_{RT})_{(extrap)}. \tag{7.143}$$

The velocity components at the boundary can then be determined from

$$u_B = n_1 V_{RNB} + t_1 V_{RTB} + (x_\tau)_B \tag{7.144}$$

and

$$v_B = n_2 V_{RNB} + t_2 V_{RTB} + (y_\tau)_B. \tag{7.145}$$

The second numerical boundary condition is imposed by extrapolating another thermodynamic variable such as entropy or total enthalpy,

$$s_B = s_{(extrap)} \tag{7.146}$$

or

$$(h_t)_B = (h_t)_{(extrap)}. \tag{7.147}$$

For steady, inviscid flows in which the total enthalpy is constant, the second numerical boundary condition can be imposed from the freestream total enthalpy as

$$(h_t)_B = (h_t)_\infty. \tag{7.148}$$

The third numerical boundary condition is imposed by computing the pressure at the boundary, p_B. The first approach is to use an extrapolation,

$$p_B = p_{(extrap)}. \tag{7.149}$$

A zero-order extrapolation works well; however, the effects on the pressure field near the boundaries are obvious. A first-order extrapolation works well for flows without

discontinuities and flows in which the pressure does not vary greatly normal to the boundary. The extrapolations across discontinuities may result in the computation of nonphysical pressures.

Another approach for determining the pressure at the boundary is to use a one-sided discretization of the compatibility relations. Starting from the compatibility relation expressed at the wall and using the continuity equation and the isentropic pressure variation, the result is

$$\frac{\partial p}{\partial n} = \kappa \rho_B V_{RTB}^2. \tag{7.150}$$

This is essentially the normal projection of the momentum equation at the wall. The κ is the curvature of the wall defined as

$$\kappa = \frac{d\hat{t}}{ds} = \frac{\pm (x_s y_{ss} - x_{ss} y_s)}{(x_s^2 + y_s^2)^{3/2}}, \tag{7.151}$$

where the + sign applies at the upper boundary and − applies at the lower boundary. If the mesh is orthogonal to the body, then $\frac{\partial p}{\partial n}$ can be approximated as a finite-difference along the η-coordinate mesh line.

A one-sided, second-order difference from the right accounting for non-uniform mesh spacing normal to the wall is of the form

$$\frac{\partial p}{\partial n} = C_1 p_B + C_2 p_{B+1} + C_3 p_{B+2} \tag{7.152}$$

where

$$C_1 = - \frac{(\Delta s_1 + \Delta s_2)^2 - \Delta s_1^2}{(\Delta s_1 + \Delta s_2) \Delta s_1 \Delta s_2},$$

$$C_2 = \frac{\Delta s_1 + \Delta s_2}{\Delta s_1 \Delta s_2},$$

$$C_3 = - \frac{\Delta s_1}{(\Delta s_1 + \Delta s_2) \Delta s_2},$$

$$\Delta s_1 = s_{B+1} - s_B,$$

and

$$\Delta s_2 = s_{B+2} - s_{B+1}.$$

A one-sided, second-order difference from the left is of the form

$$\frac{\partial p}{\partial n} = -\left(C_1 p_B + C_2 p_{B-1} + C_3 p_{B-2} \right) \tag{7.153}$$

where C_1, C_2, and C_3 are of the same form as above but with

$$\Delta s_1 = s_B - s_{B-1},$$

and

$$\Delta s_2 = s_{B-1} - s_{B-2}.$$

Using the expressions, one can obtain a relation for computing p_B.

Thus the density and specific internal energy can be determined from

$$\rho_B = \left(p_B / s_B \right)^{1/\gamma} \tag{7.154}$$

and

$$e_B = p_B / \rho_B \, (\gamma - 1). \tag{7.155}$$

With ρ_B, u_B, v_B, and e_B known, the solution at the boundary is known.

The viscous wall boundary condition

The physical boundary condition for the viscous wall is indentical to equation (7.141) for the physical boundary condition for the solid, inviscid wall. It is again assumed that there is no mass flow through the wall and so one physical boundary condition becomes

$$V_{RNB} = 0 \tag{7.156}$$

and

$$\lambda_1 = \lambda_2 = 0,$$

and

$$\lambda_3 > 0 \quad \text{and} \quad \lambda_4 < 0.$$

However, for the viscous wall boundary conditions, we now specify the zero convective eigenvalues to denote either numerical or physical boundary conditions. However, there remains at least one numerical boundary condition.

The one required numerical boundary condition is imposed through equation (7.150). This procedure has been discussed in the previous section.

A physical boundary condition is imposed by specifying the relative tangential velocity at the boundary. For viscous flow, the fluid particle adheres to the surface; thus

$$\vec{V}_{RTB} = 0. \tag{7.157}$$

The velocity components u_B and v_B can then be computed from equations (7.144) and (7.145).

The final boundary condition is imposed by a condition on the thermal behavior at the boundary. A physical boundary condition can be imposed by specifying the temperature at the boundary. Thus

$$T_B = T_{wall}. \tag{7.158}$$

A numerical boundary condition can be imposed by specifying the boundary to be adiabatic. This requires requires that

$$\frac{\partial T}{\partial n} = 0. \tag{7.159}$$

The difference formulas of equations (7.152) and (7.153) can be used to form an expression for T_B.

Thus p_B, T_B, u_B, and v_B have been determined and the solution, U, can therefore be determined.

For the external flow, viscous flat plate flow problems presented below, the problem is made transient by initially specifying

$$\vec{V}_{RTB} = \vec{V}_{RTB\infty} \tag{7.160}$$

where $\vec{V}_{RTB\infty}$ is the relative tangential velocity of the freestream flow. This velocity is then decelerated to a zero velocity through a cubic spline curve with zero slopes at the endpoints,

$$\vec{V}_{RTB}(t) = \vec{V}_{RTB\infty} \left(1 - 3q^2 + 2q^3 \right) \tag{7.161}$$

where

$$q = (t - t_0) / t_{dec}.$$

The t_0 and t_{dec} are the start time and the time interval for the transistion, respectively.

CHAPTER 8. THE NUMERICAL PROCEDURES FOR THE COUPLED DYNAMICALLY ADAPTIVE MESH METHOD

This chapter discusses the coupling of the mesh, mesh speed, and flow equations for an explicit, dynamically adaptive mesh method.

The Mesh Conservation Law

The explicit schemes presented above solve for the vector of generalized conservative variables $\hat{U}$. The vector of conservative variables is computed simply by

$$U_{i,j}^{n+1} = \hat{U}^{n+1} / V_{i,j}^{n+1}. \tag{8.1}$$

However, this requires knowing $V_{i,j}^{n+1}$.

The $V_{i,j}^{n+1}$ is computed from the mesh conservation law as discussed in references [41] and [24]. The mesh conservation law is derived from the time integration equations with the assumption of a uniform solution for the conservative variables. For the explicit, two-stage Lax-Wendroff method, the mesh conservation law can be expressed in two stages as

$$V_{i,j}^{*} = V_{i,j}^{n} - \Delta\tau \, \hat{Z}_{i,j}^{n}, \tag{8.2}$$

$$V_{i,j}^{**} = V_{i,j}^{*} - \Delta\tau \, \hat{Z}_{i,j}^{*} \tag{8.3}$$

and

$$V_{i,j}^{n+1} = \frac{1}{2} \left\{ V_{i,j}^{n} + V_{i,j}^{**} \right\} \tag{8.4}$$

where $\hat{Z}$ is the vector sum of speeds of the faces of the finite-volume.

The equation makes physical sense. It states that the change in volume is due to the motion of the cell faces.

The Integration of the Mesh Speeds

During each of the stages of the explicit scheme, the mesh can be integrated in time using the same method as for the flow equations.

For the explicit, two-stage Lax-Wendroff method, the time integration of the mesh is

$$\vec{r}_{i,j}^{*} = \vec{r}_{i,j}^{n} + \Delta\tau \, \vec{z}_{i,j}^{n}, \tag{8.5}$$

$$\vec{r}_{i,j}^{**} = \vec{r}_{i,j}^{*} + \Delta\tau \, \vec{z}_{i,j}^{*} \tag{8.6}$$

and

$$\vec{r}_{i,j}^{n+1} = \frac{1}{2} \left\{ \vec{r}_{i,j}^{n} + \vec{r}_{i,j}^{**} \right\}. \tag{8.7}$$

Coupling

The approach for coupling the flow and mesh equations to advance the solution and mesh with second-order accuracy in space and time is a multi-step procedure. The flow equations are affected by the mesh motion through the mesh speed terms. The mesh equations are affected by the flow through the W weight function and the time difference of the W showing up in the mesh speed equations. When flow or mesh

data are needed when the solution or mesh is not known, that quantity is lagged or linearized. Second-order time accuracy is possible through a two-stage procedure.

The Explicit, Multi-Stage Dynamically Adaptive Mesh Method

The steps in the method include:

1) An initial mesh and solution are specified. It is assumed that the initial mesh satisfies the mesh equations and that the initial solution is consistent with the mesh.

2) Using the initial mesh and solution, the mesh speed equation is solved to determine the initial mesh speeds. If the mesh speeds are computed from a backwards time-difference of the mesh, then the initial mesh speeds are set to zero.

3) The time step, $\Delta\tau$, is computed using the CFL condition or it is specified.

4) The first stage is carried out to compute $\hat{U}^*$,

$$\hat{U}^*_{i,j} = \hat{U}^n_{i,j} - \Delta\tau\ \hat{R}^n_{i,j}.$$

5) The mesh conservation law is solved for the volume at the first stage,

$$V^*_{i,j} = V^n_{i,j} - \Delta\tau\ \hat{Z}^n_{i,j}.$$

6) The flow solution is computed for the first stage,

$$U^*_{i,j} = \hat{U}^*_{i,j} \ / \ V^*_{i,j}.$$

The flow boundary conditions are also computed for the first-stage solution.

7) The first-stage algorithm is applied to the integration of the mesh speeds to obtain the mesh at the first stage,

$$\vec{r}^{\,*}_{i,j} = \vec{r}^{\,n}_{i,j} + \Delta\tau\ \vec{z}^{\,n}_{i,j}.$$

Since the mesh has changed, the volume and cell-face area vectors are recomputed.

8) The mesh equation is solved for the first-stage conditions. This step may not be needed if the mesh speed equations are to be solved.

9) The mesh speed equation is solved for the first-stage conditions. If the mesh speeds are computed from a backwards time difference, then the difference is performed.

10) The second stage is carried out to compute $\hat{U}^{**}$,

$$\hat{U}^{**}_{i,j} = \hat{U}^{*}_{i,j} - \Delta\tau\ \hat{R}^{*}_{i,j}.$$

11) The finite-volume solution $\hat{U}^{n+1}_{i,j}$ is computed

$$\hat{U}^{n+1}_{i,j} = \frac{1}{2}\left[\,\hat{U}^{n}_{i,j} + \hat{U}^{**}_{i,j}\,\right].$$

12) The mesh conservation law is solved for the volume at the second stage,

$$V^{**}_{i,j} = V^{*}_{i,j} - \Delta\tau\ \hat{Z}^{*}_{i,j}.$$

13) The volume at the $n+1$ time level is computed

$$V^{n+1}_{i,j} = \frac{1}{2}\left(\,V^{n}_{i,j} + V^{**}_{i,j}\,\right).$$

14) The flow solution is computed for the $n+1$ time level,

$$U^{n+1}_{i,j} = \hat{U}^{n+1}_{i,j} \,/\, V^{n+1}_{i,j}.$$

The flow boundary conditions are also computed for the second stage solution.

15) The second-stage algorithm is applied to the integration of the mesh speeds to obtain the mesh at the $n+1$ time level, $\vec{r}^{\,n+1}$,

$$\vec{r}_{i,j}^{\,n+1} = \vec{r}_{i,j}^{\,n} + \frac{1}{2}\,\Delta\tau\,\left(\,\vec{z}_{i,j}^{\,n} + \vec{z}_{i,j}^{\,*}\,\right).$$

Since the mesh has changed, the volume and cell-face area vectors are recomputed.

16) The mesh equation is solved for the $n+1$ time level conditions. This step may not be needed if the mesh speed equations are to be solved.

17) The mesh speed equation is solved for the $n+1$ time level conditions. If the mesh speeds are computed from a backwards time difference, then the difference is performed.

18) If the final time has not been reached, go to step 3 and repeat the procedure.

CHAPTER 9. THE RESULTS OF NUMERICAL EXPERIMENTS WITH THE MESH AND MESH SPEED EQUATIONS

The behavior of the numerical solution procedures for the mesh and mesh speed equations is demonstrated through a series of numerical experiments involving simple domain geometries, which allow some aspect of the mesh and mesh speed equation solutions to be known prior to the computations. Thus an evaluation of the performance of the mesh and mesh speed equation solution procedures can be determined.

Experiments with the Mesh Equations

The first set of experiments involved a square domain with sides of unit length. This is perhaps the simplest of domain geometries. For mesh points spaced equally along the boundaries and for a uniform W function, the exact solution for the mesh equation for all values of λ_S, λ_O, and λ_A is a rectangular mesh. This is also the solution when orthogonality boundary conditions are enforced.

For a (3x3) mesh for which $NI = 3$ and $NJ = 3$ in which Dirichlet boundary conditions are enforced, only the center point $(i = 2, j = 2)$ coordinates needs to be computed. The computations demonstrated that only two iterations of the mesh equations were required to reach the exact solution regardless of the values of λ_S, λ_O, and λ_A when $\omega_G = 1.0$. When the value of ω_G was not unity, more iterations

were required.

For a (5x5) mesh, the solution path of the mesh equation is dependent on the values of λ_S, λ_O, λ_A, and ω_G. However, the correct final solution is still a rectangular mesh. Figure 9.1(a) shows the initial mesh for which the mesh points were chosen in a fairly random manner while not allowing mesh lines to cross or fall outside the domain boundaries. Table 9.1 summarizes the performance of the mesh equation solution procedure. The iteration procedure was performed until the residual fell below a value of $1.0x10^{-7}$. The residual is defined by the average magnitude of the change in the locations of the mesh points. The $*$ in Table 9.1 indicates that a solution could not be computed. The $-$ indicates that the parameter is not applicable. The values of ω_G and ω_{GB} presented in the table are the values for which the least number of iterations were required or the maximum values for which a stable solution could be computed. Table 9.1 also lists the value of the Lagrangian integral, I, for the initial and final meshes. A trapezoidal integration is used to numerically compute I. Note that the values of I were reduced for all cases, indicating that the extremum value of the integral was computed. Figure 9.1(b) shows the final, rectangular mesh.

The next set of experiments involved a quadrilateral domain with the bottom and top sides kinked into two straight-line segments. This presents a more complex domain geometry. Since the boundary geometry is symmetric with respect to the $x = 0.5$ line, the mesh solution should also be symmetric. Figure 9.2(a) shows the initial mesh. Table 9.2 summarizes the performance of the mesh equation solution procedure.

It was possible to obtain a mesh solution for the case in which only the orthogonality measure was driving the mesh solution and Dirichlet boundary conditions were

Table 9.1: Mesh solutions for the (5x5) mesh on a square domain

Dirichlet Boundary Conditions							
λ_S	λ_O	λ_A	ω_G	ω_{GB}	Iterations	Initial I	Final I
1.0	0.0	0.0	1.00	-	19	20.1201	18.0000
0.0	1.0	0.0	0.90	-	43	0.4954	0.0000
0.0	0.0	1.0	1.00	-	30	14.5275	9.0000
1.0	1.0	1.0	1.00	-	19	35.1429	27.0000
1.0	1.0	1.0	1.20	-	11	35.1429	27.0000

Orthogonality Boundary Conditions							
λ_S	λ_O	λ_A	ω_G	ω_{GB}	Iterations	Initial I	Final I
1.0	0.0	0.0	1.00	0.80	51	20.1201	18.0000
0.0	1.0	0.0	0.05	0.05	*	0.4954	*
0.0	0.0	1.0	1.00	0.80	125	14.5275	9.0000
1.0	1.0	1.0	1.00	0.80	81	35.1429	27.0000
1.0	1.0	1.0	1.90	0.80	37	35.1429	27.0000

enforced. Significant underrelaxation was required to obtain the solution. A solution was not possible when orthogonality boundary conditions were enforced. This behavior supports the fact that the orthogonality integral equations are a degenerate set of equations.

It was possible to obtain a mesh solution for the case in which only the volume adaption measure was driving the mesh solution. For Dirichlet boundary conditions in which the mesh points on the boundaries were evenly spaced, the final mesh formed finite-volumes of equal volume. This is seen in Figure 9.2(b).

The next experiments involved defining the initial mesh so that some of the mesh points were located outside of the domain boundary and that some of the mesh lines crossed. The purpose was to check how robust the mesh equation solution procedure was to the initial solution. A boundary value problem is dependent only on the boundary values and should be able to compute a solution independent of the

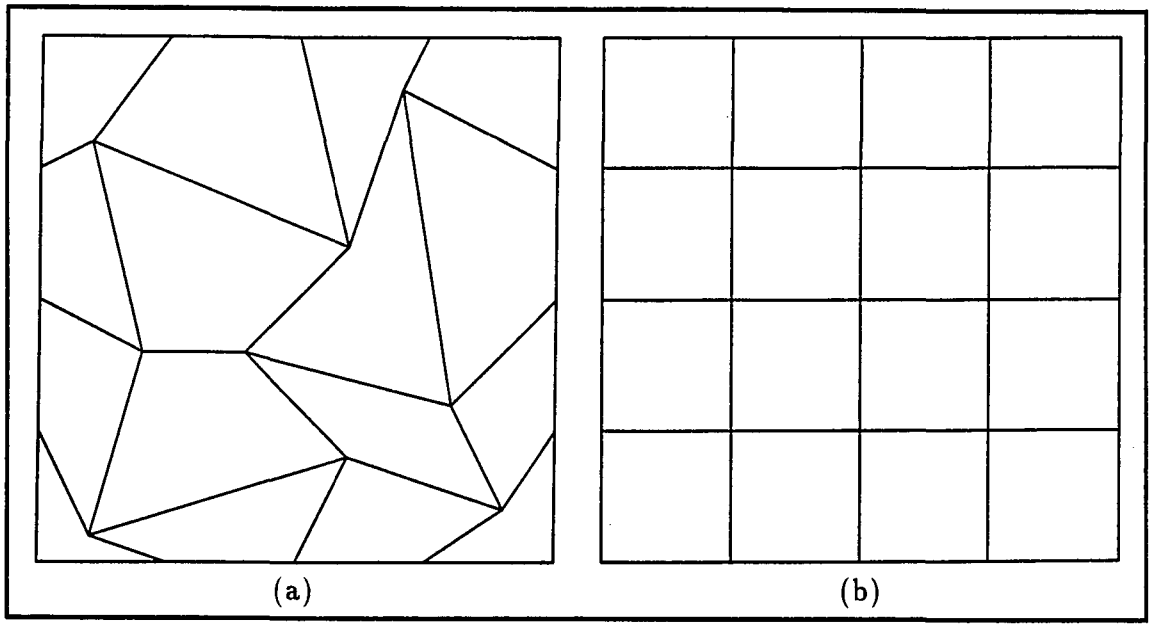

Figure 9.1: The (a) initial and (b) final (5x5) mesh for the square domain

values on the interior. Table 9.3 shows the performance of the mesh equation solution procedure for Dirichlet boundary conditions. It was not possible to obtain solutions when the orthogonality boundary conditions were enforced. The last row in Table 9.3 shows the optimum ω_G for the iteration. One observation is that values of ω_G above the optimum produced possible unstable iterations. It was observed that the smoothness measure was needed to obtain a stable solution. This is consistent with mathematical character of the mesh equations.

The next set of experiments involved the geometry for the converging-diverging nozzle examined in the next chapter. Figure 9.3 shows the initial (31x8) mesh obtained from equally spacing the mesh point on the boundary and then using a transfinite interpolation to compute the interior mesh. Table 9.4 presents the performance of the mesh equation solution procedure when the orthogonality boundary condi-

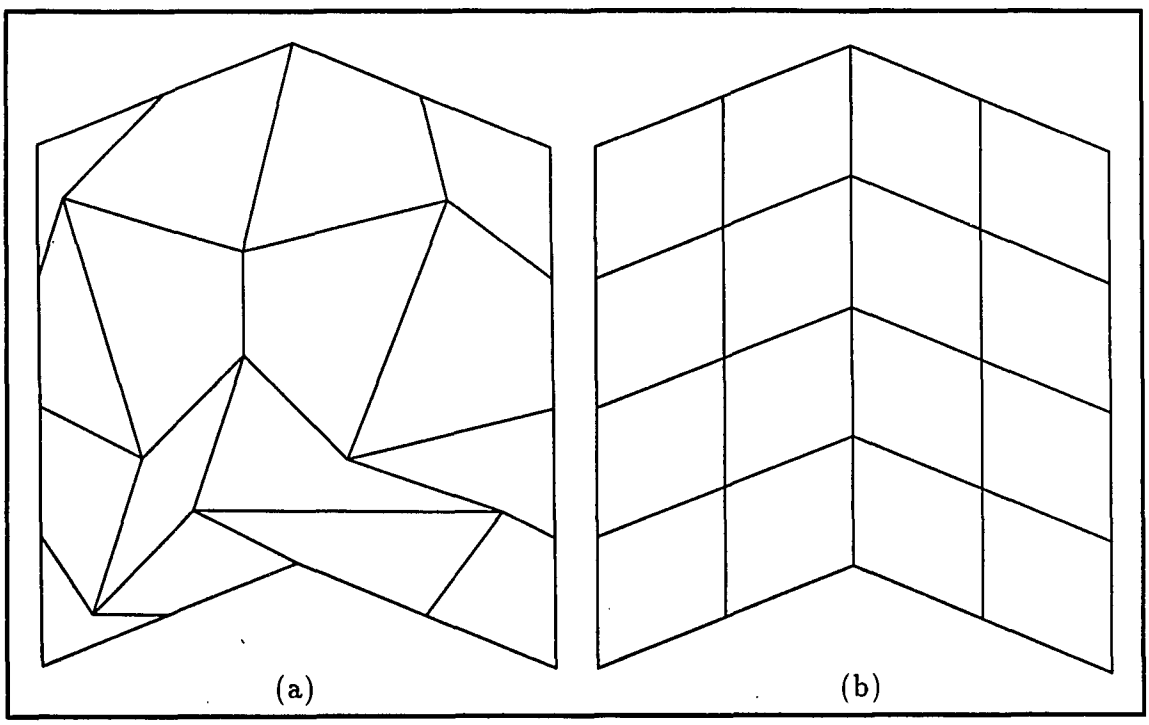

Figure 9.2: The (a) initial (5x5) mesh for the kinked quadrilateral domain and (b) final mesh for $(\lambda_S, \lambda_O, \lambda_A) = (0.0, 0.0, 1.0)$ and Dirichlet boundary conditions

tions are enforced. The CPU indicates the amount of CPU time require on the NASA Lewis Cray XMP. Figure 9.4 shows the final meshes for various values of the mesh parameters.

When one considers using one of these meshes in the computation of the flowfield in the converging-diverging nozzle, it would be desirable to choose a mesh in which mesh points were clustered in the throat region, where the greatest solution gradients would be located. This appears to be provided by the mesh generated using the parameters $(\lambda_S, \lambda_O, \lambda_A) = (1, 1, 0)$ as shown in Figure 9.4(d). The computation of this mesh is represented by the last line of Table 9.4. The values of ω_G and ω_{GB} are

Table 9.2: Mesh solutions for the (5x5) mesh on a kinked quadrilateral domain

Dirichlet Boundary Conditions							
λ_S	λ_O	λ_A	ω_G	ω_{GB}	Iterations	Initial I	Final I
1.0	0.0	0.0	1.00	-	19	20.6813	18.3910
0.0	1.0	0.0	0.40	-	127	0.6264	0.0082
0.0	0.0	1.0	1.00	-	29	14.2219	9.0000
1.0	1.0	1.0	1.00	-	18	35.5297	27.5304
1.0	1.0	1.0	1.20	-	14	35.5297	27.5304

Orthogonality Boundary Conditions							
λ_S	λ_O	λ_A	ω_G	ω_{GB}	Iterations	Initial I	Final I
1.0	0.0	0.0	1.00	1.00	83	20.6813	18.1070
0.0	1.0	0.0	0.05	0.05	*	0.6264	*
0.0	0.0	1.0	1.00	0.80	99	14.2219	8.8485
1.0	1.0	1.0	1.00	0.80	195	35.5297	26.6283
1.0	1.0	1.0	1.70	0.80	132	35.5297	26.6283

presented in Table 9.4 are the values which required the least number of iterations. Values of ω_G above 2.0 resulted in an unstable solution.

Note that when λ_A is not zero, the mesh generation procedure attempts to equidistribute the cell areas. This has the effect of increasing the longitudinal dimension of the cells in the throat region. This can be seen in Figures 9.4(b) and 9.4(c).

Table 9.3: Mesh solutions for the (5x5) mesh for the kinked quadrilateral domain with Dirichlet boundary conditions with an overlapping initial mesh

λ_S	λ_O	λ_A	ω_G	Iterations
1.0	0.0	0.0	1.0	21
0.0	1.0	0.0	0.4	*
0.0	0.0	1.0	1.0	*
1.0	1.0	1.0	1.0	34
1.0	1.0	1.0	1.1	17

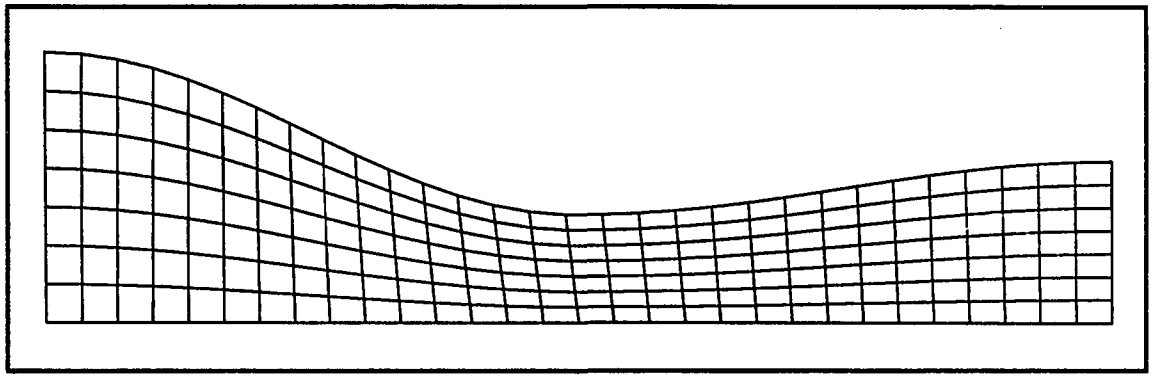

Figure 9.3: The initial (31x8) mesh for the CD nozzle geometry

Table 9.4: Mesh solutions for the (31x8) mesh for the converging-diverging nozzle with orthogonality boundary conditions enforced

λ_S	λ_O	λ_A	ω_G	ω_{GB}	Iterations	Initial I	Final I	CPU
1.0	0.0	0.0	1.0	1.0	4173	413.5888	400.3292	9.523
0.0	1.0	0.0	0.1	0.1	*	5.0686	*	*
0.0	0.0	1.0	1.0	1.0	8971	185.2906	173.8335	20.455
1.0	1.0	1.0	1.0	1.0	4431	603.9481	595.8567	10.104
1.0	1.0	0.0	1.0	1.0	7521	418.6574	400.5390	17.121
1.0	1.0	0.0	2.0	1.2	1226	418.6574	400.5390	2.855

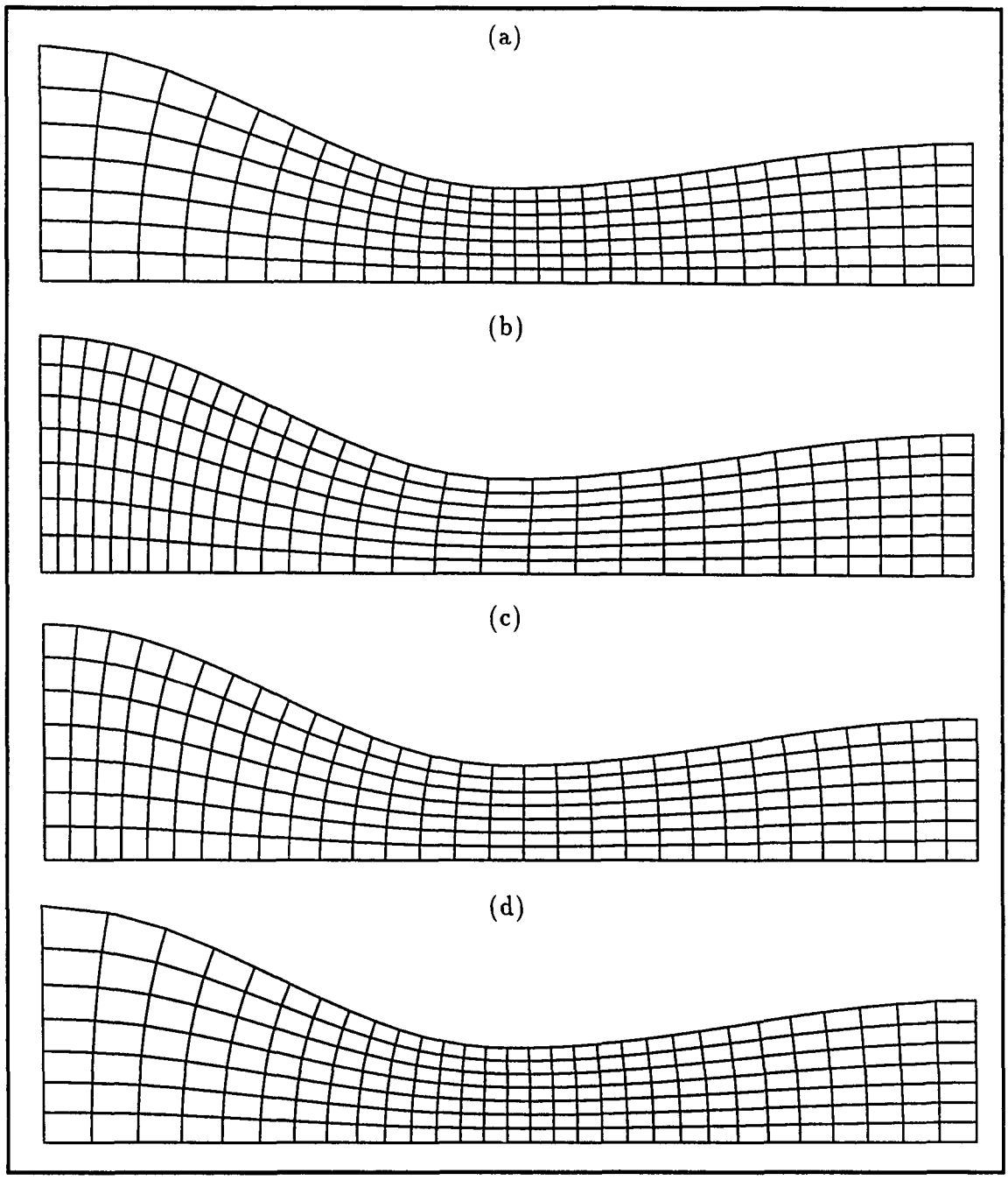

Figure 9.4: The final (31x8) meshes for the converging-diverging nozzle geometry for various values of $(\lambda_S, \lambda_O, \lambda_A)$: (a) (1,0,0); (b) (0,0,1); (c) (1,1,1); (d) (1,1,0)

Experiments with the Mesh Speed Equations and Dynamic Boundaries

The performance of the mesh speed equations is demonstrated for a series of numerical experiments in which the motion is specified on the boundary. The cases are summarized in Table 9.5 which also lists the values of the relaxation parameters ω_G and ω_{GS} used for each computation. All cases started with a unit square domain with a square mesh of dimension (11x11). The W function was set to unity. The time integrations were performed for 50 time steps with a time step of $\Delta\tau = 0.02$ seconds to a final time of $t = 1.0$ seconds. The mesh parameters were $(\lambda_S, \lambda_O, \lambda_A)$ = (1.0,1.0,1.0) and $\lambda_C = 30.0$. The iteration tolerance for the point relaxation was $1.0x10^{-8}$. Throughout the computations, $\omega_{GB} = 1.0$ and $\omega_{GSB} = 1.0$. Table 9.6 summarizes the overall results for each case. The total number of iterations of the mesh speed equations is listed. The errors are the differences in mesh point locations between the final mesh obtained from the time integration and the mesh satisfying the mesh equations at the final time. The errors are expressed as percentages of the length of the uniform mesh spacing.

Case $BM1$ (boundary motion case 1) involved specifying the right boundary to move to the right at a rate of 1.0 units per second. The right boundary moved from a location of $x = 1.0$ units to a location of $x = 2.0$ units during the time integration.

Table 9.5: The cases involving dynamic boundaries

Case	Description	ω_G	ω_{GS}
$BM1$	Right Boundary Pulled Right	1.0	1.0
$BM2$	Square Mesh Rotated	1.0	1.0
$BM3$	Lower Boundary Rotated	1.0	1.0
$BM4$	Upper-Right Corner Pulled	1.7	1.7

Table 9.6: The overall results for the experiments involving dynamic boundaries

Case	Total # of Iterations	(%) Avg Error	(%) Max Error
$BM1$	107	0.0000	0.0000
$BM2$	20650	0.0000	0.0000
$BM3$	17308	0.0396	0.8176
$BM4$	5989	0.3046	5.4933

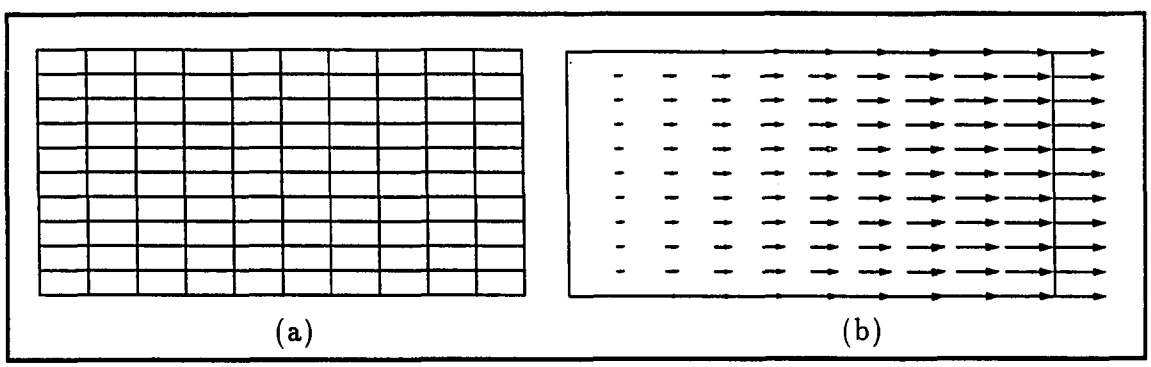

Figure 9.5: The (a) mesh and (b) mesh speeds at $t = 1.0$ seconds for the square being pulled at the right boundary (case BM1)

The left boundary was held fixed. On the upper and lower boundaries, the mesh speeds were linearly interpolated between the zero speed at the left boundary and the prescribed speed at the right boundary. The mesh speeds were computed at the first time step and remain constant throughout the time integration. The mesh stretches in the horizontal direction and remained rectangular. The mesh and the mesh speeds at the final time are shown in Figure 9.5.

Case $BM2$ involved specifying the mesh speeds on the boundary such that the square mesh rotated about the lower left corner at a rate of 90 degrees/second. At the final time, the mesh was still square and rotated 90 degrees. The magnitudes of

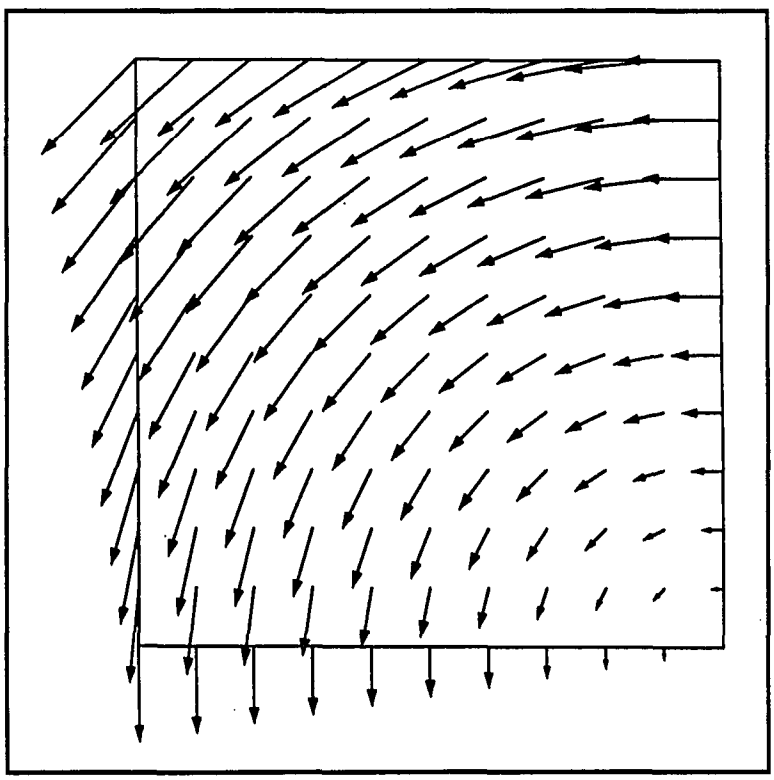

Figure 9.6: The mesh speeds at $t = 1.0$ seconds for the unit square being rotated 90 degrees/second (case BM2)

the mesh speeds remain constant; however, the direction of the mesh speeds changed. Figure 9.6 shows the mesh speeds at the final time.

Case $BM3$ involved specifying the lower boundary to rotate about the lower left corner at a rate of 20 degrees/second. At the final time, the lower boundary was rotated 20 degrees. This tests the ability of the method to maintain mesh orthogonality at the boundary. The left and upper boundaries were held fixed. The mesh speeds at the right boundary were computed from a linear interpolation between the mesh speed at the lower right corner and the zero mesh speed at the upper right corner.

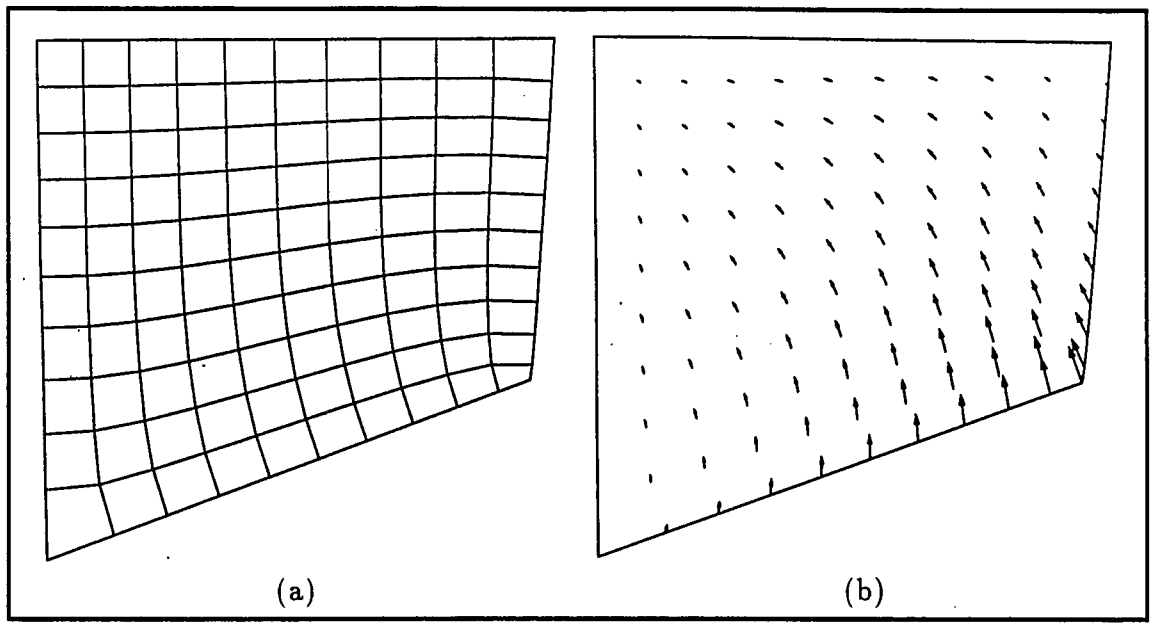

Figure 9.7: The (a) mesh and (b) mesh speeds at $t = 1.0$ seconds for the unit square with the lower boundary being rotated at 20 degrees/second (case BM3)

The final mesh satisfied the mesh equations very well. Figure 9.7 shows the mesh and mesh speeds at the final time.

The effectiveness of the mesh control law damping factor, λ_C, was investigated for case BM3 and the results are presented in Table 9.7. Figure 9.8 shows a reduction in the percentage of the average error for increasing values of λ_C. The fact that the curve bottoms out indicates that one should be able to pick a high value of λ_C and obtain a reduction in error without having to search for an optimum value. Along with the errors, the table presents the number of iterations of the mesh equations required at the final time to correct the final integrated mesh and the total number of iterations of the mesh speed equations required in computing the mesh speeds. While the errors decreased, the equations became stiffer, and so required more iterations of

Table 9.7: The effectiveness of the mesh control law damping factor, λ_C

λ_C	Mesh Iterations	(%) Avg Error	(%) Max Error	Total # of Iterations
0.0	245	0.5498	11.2154	12242
5.0	217	0.1873	3.8047	15120
10.0	202	0.1074	2.1858	15928
30.0	177	0.0396	0.8176	17308
35.0	173	0.0343	0.7111	17629
40.0	170	0.0304	0.6523	17971
50.0	165	0.0249	0.6578	18975
60.0	161	0.0213	0.6638	20127
70.0	158	0.0189	0.6718	21331

the mesh speed equations. This has to be considered when the value of λ_C is chosen.

The effectiveness of the relaxation parameter was investigated for case BM3. It was determined that a value of $\omega_{GS} = 1.8$ reduced the total number of iterations of the mesh speed equations to a minimum of 4309 iterations.

Case $BM4$ involved specifying the mesh velocity of the upper right mesh point to be $x_T = 1.0$ units per second and $y_T = 1.0$ units per second. The left and lower boundaries were stationary. The mesh speeds along the upper and right boundaries were computed from a linear interpolation between a zero mesh speed and the mesh speed of the upper right corner. The final position of the upper right corner point was $x = 2.0$ units and $y = 2.0$ units. The final mesh satisfied the mesh equations very well. Figure 9.9 shows the mesh and mesh speeds at the final time.

The sensitivity of the mesh speed iteration procedure to the value of the iteration convergence tolerance was investigated for case BM4. It was found that when the tolerance was reduced from $1.0x10^{-9}$ to $1.0x10^{-3}$, the total number of iterations required for the mesh speed equations dropped from 7667 to 170 while the error

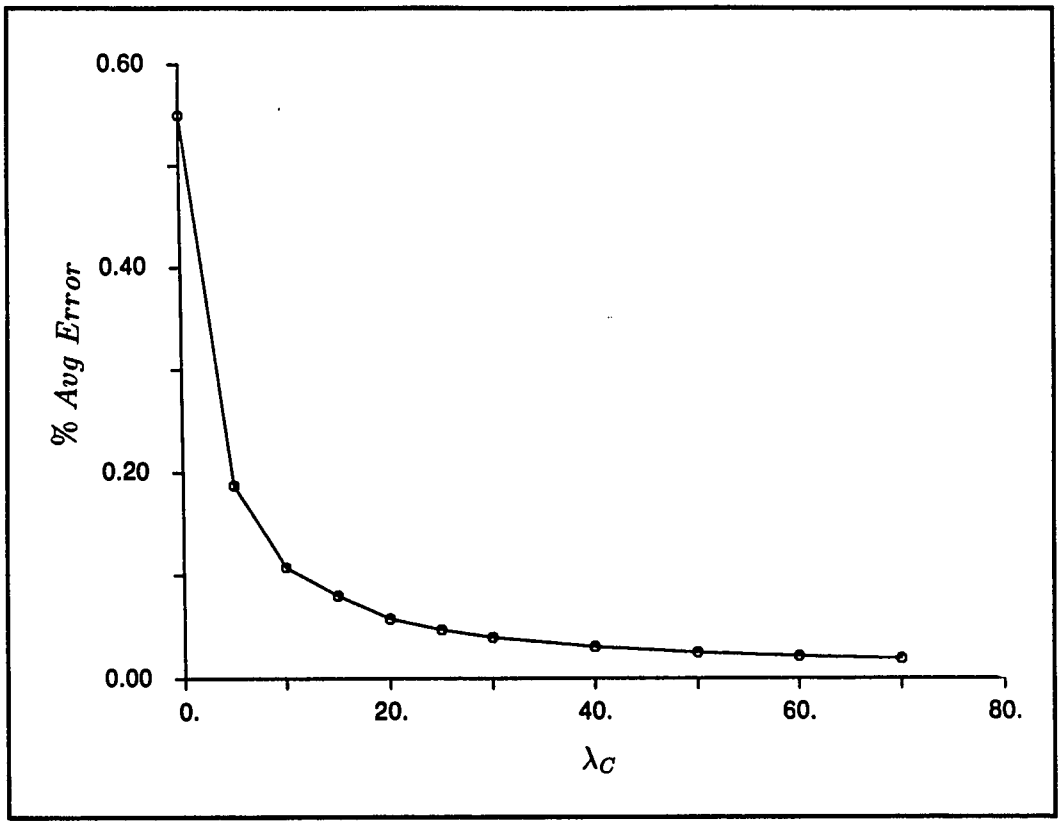

Figure 9.8: The effectiveness of the λ_C damping factor

in the final mesh remained essentially unchanged. This suggests that the iteration procedure for the mesh speed equations is quite robust with a rapid initial rate of convergence. This leads to a potential for a substantial savings of computational effort.

The effects of varying λ_S, λ_O, and λ_A were also investigated for case BM4. Figure 9.10 shows the final mesh for the cases in which $(\lambda_S, \lambda_O, \lambda_A) = (1.0, 1.0, 0.0)$ and $(\lambda_S, \lambda_O, \lambda_A) = (0.25, 0.0, 1.0)$. It should be evident that λ_A is useful in controlling the variation of the cell volumes.

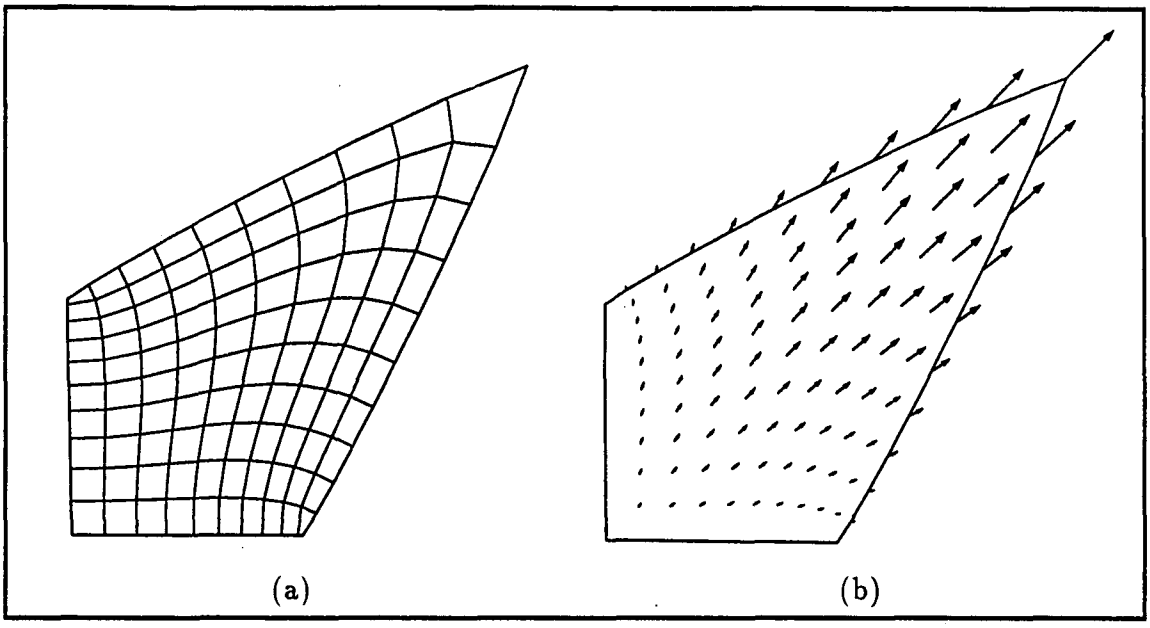

Figure 9.9: The (a) mesh and (b) mesh speeds at $t = 1.0$ seconds for the unit square being pulled at the upper right corner (case BM4)

Experiments with the Mesh Speed Equations and Dynamic Solutions

These numerical experiments demonstrate the performance of the method when the solution on the mesh is dynamic and the mesh adapts to the solution dynamics.

Case $DS1$ (dynamic solution case 1) involved specifying a quadratic weighting

Table 9.8: The cases involving dynamic solutions

Case	Description	ω_G	ω_{GS}
$DS1$	Quadratic $W(\xi, \eta, \tau)$	1.7	1.7
$DS2$	Lower Boundary Rotated with $W(\xi, \eta, \tau)$	1.7	1.7
$DS3$	Exponential $u(x, y, t)$	1.7	1.7
$DS4$	Cylindrical Discontinuity	1.0	1.0
$DS5$	Quadratic Boundary Layer	1.5	1.5

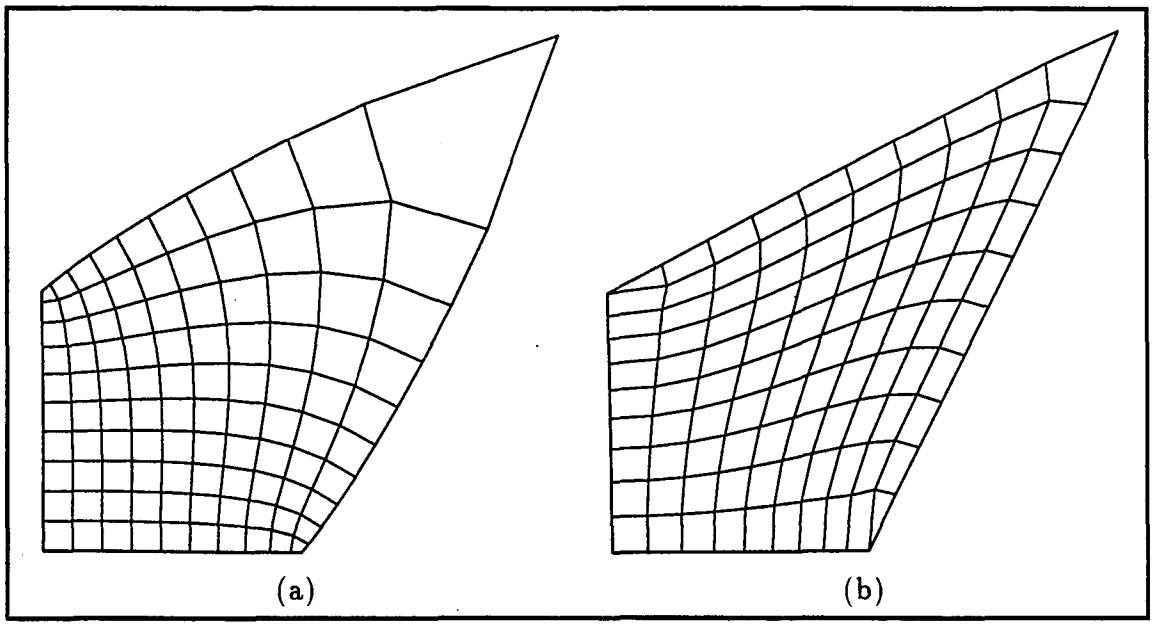

Figure 9.10: The meshes at $t = 1.0$ seconds for the unit square being pulled at the upper right corner (case BM4) with (a) $(\lambda_S, \lambda_O, \lambda_A) = (1.0, 1.0, 0.0)$ and (b) $(\lambda_S, \lambda_O, \lambda_A) = (0.25, 0.0, 1.0)$

function $W(\xi, \eta, \tau)$ of the form

$$W(\xi, \eta, \tau) = \lambda_0 + \frac{\lambda_1}{3}\left(\xi^2 + \xi\eta + \eta^2\right)\tau^2 \tag{9.1}$$

where

$$0 \le \xi \le 1, \quad 0 \le \eta \le 1, \quad \text{and} \quad 0 \le \tau \le 1.$$

The W and the derivatives of W needed for the mesh and mesh speed equations can be computed exactly and allow testing of the mesh and mesh speed equations without the discretization errors associated with W.

The initial mesh was the square mesh on the unit square domain. The initial square mesh satisfied the mesh equation at $\tau = 0$ since $W(\xi, \eta, 0) = 1.0$. The mesh parameters were $(\lambda_S, \lambda_O, \lambda_A) = (1.0, 1.0, 1.0)$ and $\lambda_C = 40.0$. The mesh speeds were

Table 9.9: The overall results for the experiments involving the dynamic solutions

Case	Total # of Iterations	(%) Avg Error	(%) Max Error
$DS1$	19846	0.1182	2.5953
$DS2$	7479	0.0623	1.0277
$DS3$	25662	0.4401	1.6899
$DS4$	17868	0.8564	20.4793
$DS5$	19304	0.2475	5.1069

iterated to a residual of $1.0x10^{-8}$. Figure 9.11 shows the mesh and mesh speeds at the final time. The effect of the clustering of the mesh is expressed as the smallest mesh spacing expressed as a percentage of the uniform mesh spacing. A clustering of 7.23% was achieved. This is a considerable amount of clustering for an (11x11) mesh. Since the function is symmetric with respect to the diagonal, the mesh and mesh speeds were also symmetric.

An alternative to computing the mesh speeds and integrating to obtain the mesh is to solve the mesh equation and then compute the mesh speeds using a backwards time difference. Solving the mesh speed equations for case $DS1$ required 14.945 CPU seconds on the Cray XMP, while using a backwards time difference only required 11.141 CPU seconds. The average error was reduced from 0.1182% to 0.0044%. Solving the mesh equation at every stage and backwards differencing the mesh is more efficient; however, one should note that the backwards differencing of the mesh to obtain the mesh speeds may cause the mesh speeds to lag the physics of the solution.

Case $DS2$ involved a combined mesh dynamics of boundary motion and adaptive mesh motion. The quadratic W function is combined with the a specified rotation of

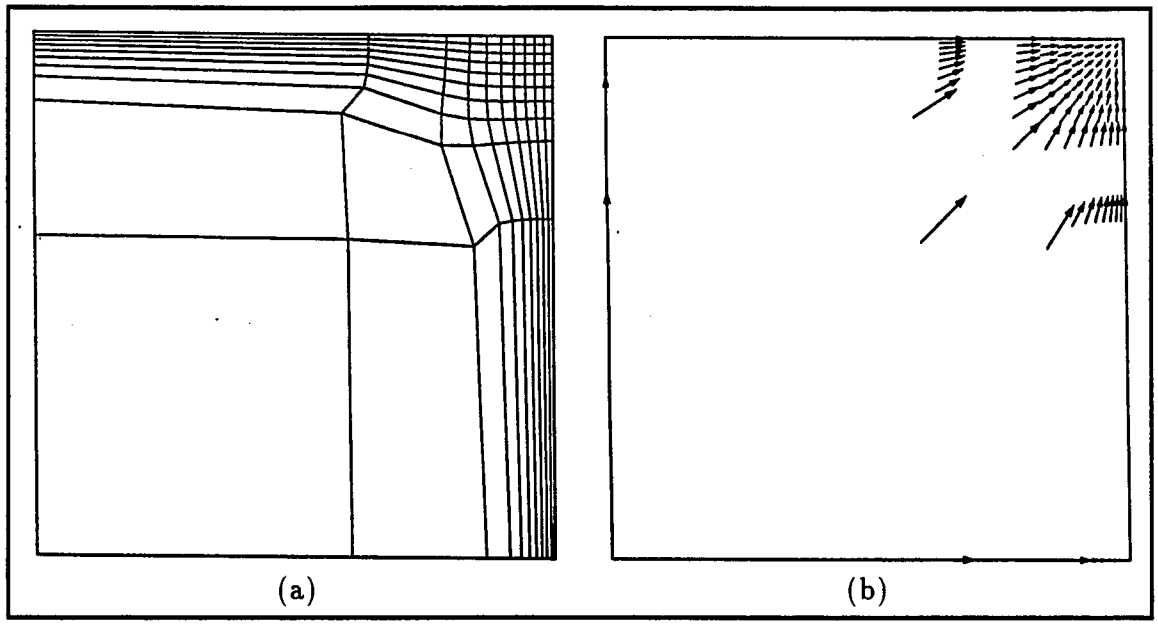

Figure 9.11: The (a) mesh and (b) mesh speeds at $t = 1.0$ seconds for the unit square mesh with a quadratic $W(\xi, \eta, \tau)$ (case DS1)

the lower boundary at a rate of 20 degrees/second. The mesh adaption parameters were $\lambda_0 = 1.0$ and $\lambda_1 = 1.0$. Figure 9.12 shows the mesh and mesh speeds at the final time. The final mesh satisfied the mesh equations very well. It was observed that if λ_1 was increased to obtain more adaption or if the lower boundary was rotated further, the mesh line near the lower right corner was forced outside of the domain due to the orthogonality boundary conditions. To prevent such behavior one could impose an angle boundary condition other than orthogonality or impose minimum and maximum limits on the movement of the mesh points along the boundary.

Case $DS3$ involved specifying an exponential function in space and quadratic in time of the form

$$u(x,y,t) = \left\{\frac{1-\exp[10(x-1)]}{1-\exp(-10)}\right\}\left\{\frac{1-\exp[10(y-1)]}{1-\exp(-10)}\right\} t^2.$$

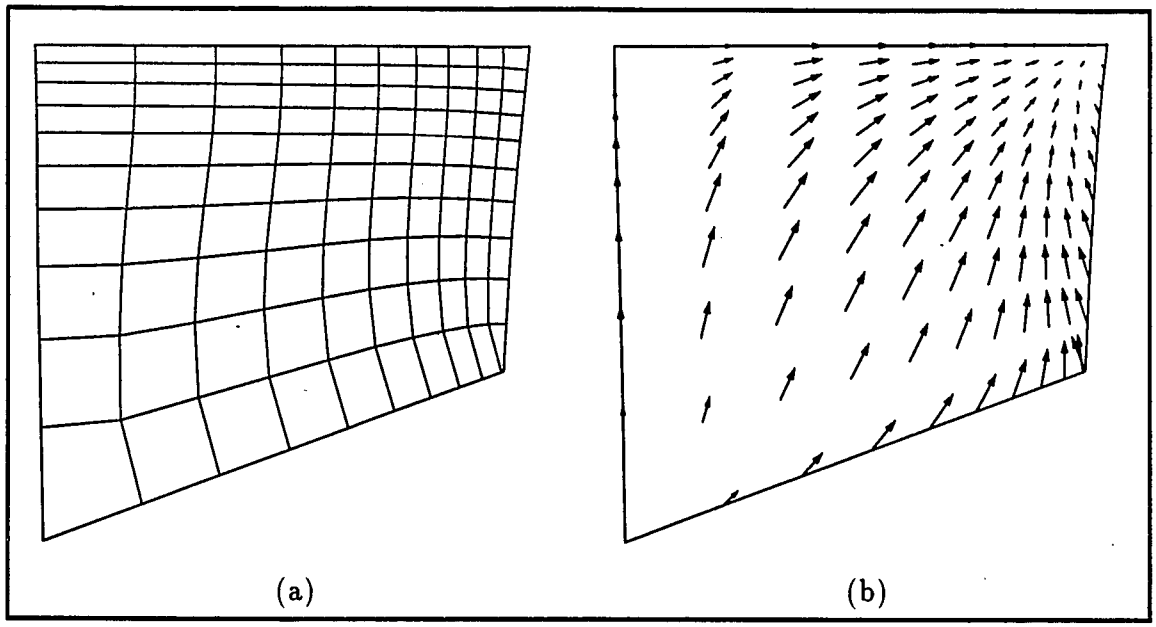

Figure 9.12: The (a) mesh and (b) mesh speeds at $t = 1.0$ seconds for the unit square with a quadratic $W(\xi, \eta, \tau)$ and the lower boundary in rotation (case DS2)

The mesh parameters were set to $(\lambda_S, \lambda_O, \lambda_A) = (1.0, 1.0, 2.0)$, $\lambda_2 = 1.0$, $\lambda_3 = 10.0$, and $\lambda_C = 40.0$. The solution was smoothed 5 times prior to computing W and its derivatives. The mesh speeds were iterated to a tolerance of $1.0x10^{-8}$. Figure 9.13 shows the mesh and mesh speeds at the final time. A clustering of 35.35% was achieved. Since the solution is symmetric with respect to the diagonal, the final mesh and mesh speeds are symmetric with respect to the diagonal.

Case $DS4$ involved specifying a cylindrical discontinuity on the square which is initially at a radius of $r = 0.25$ units and travels at a linear rate of 0.5 units per second radially from the origin. At the final time of $t = 1.0$ seconds, the discontinuity is at a radial distance of $r = 0.75$ units. The discontinuous solution has a value of 1.0 in front of the discontinuity and a value of 2.0 behind the discontinuity. The mesh

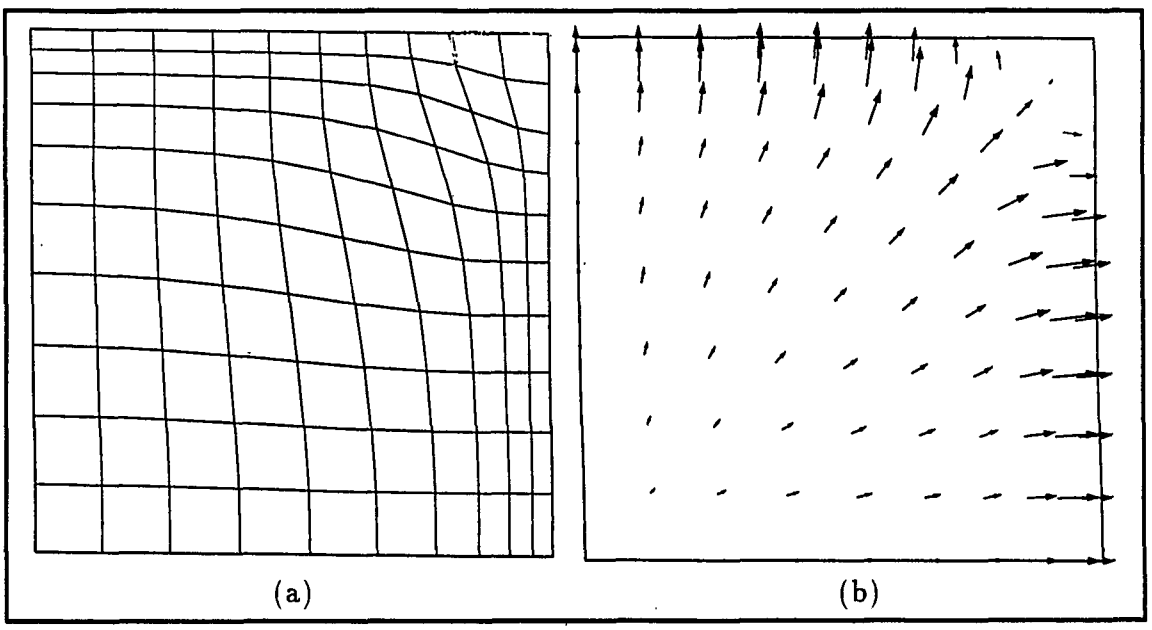

Figure 9.13: The (a) mesh and (b) mesh speeds at $t = 1.0$ seconds for the unit square with an exponential $u(x,y,t)$ (case DS3)

parameters were set to $(\lambda_S, \lambda_O, \lambda_A) = (1.0,1.0,1.0)$, $\lambda_1 = 5.0$, and $\lambda_C = 40.0$. The solution was smoothed 5 times prior to the computation of W and its derivatives. Figure 9.14 shows the mesh and mesh speeds at the final time. A clustering of 69.53% was achieved. The mesh and mesh speeds seem fairly symmetric with respect to the diagonal. Attempts at obtaining a greater amount of clustering about the discontinuity were unsuccessful. At higher values of λ_1, the iterations of the mesh speed equations became unstable. If the solution was smoothed more, the gradients were reduced. It is not known whether the mesh speed equations or the iterative numerical method is the source of the instability.

Case $DS5$ involved specifying a time-dependent, quadratic boundary layer solution on the square. At $t = 0.0$, the flow was uniform; $u(x,y,t) = 1.0$. As time

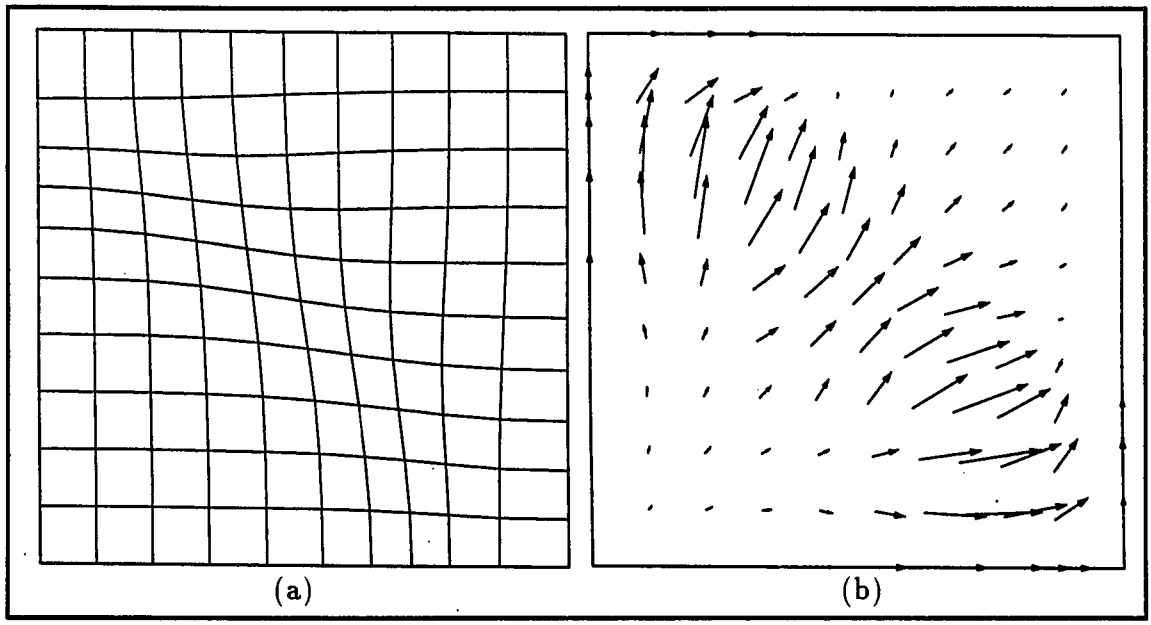

Figure 9.14: The (a) mesh and (b) mesh speeds at $t = 1.0$ seconds for the unit square with a cylindrical discontinuity (case DS4)

advanced to $t = 1.0$ seconds, the u at the wall was linearly reduced to $u(x, y = 0.0, t = 1.0) = 0.0$. A quadratic curve was used to define $u(x, y, t)$ from the wall to a boundary-layer thickness of $y = 0.02$. The mesh parameters were $(\lambda_S, \lambda_O, \lambda_A) =$ (1.0,1.0,3.0), $\lambda_1 = 500.0$, and $\lambda_C = 30.0$. The solution was smoothed 30 times prior to the computation of W and its derivatives. Figure 9.15 shows the mesh and mesh speeds at the final time. The minimum spacing was about 15.65% of the uniform spacing. The mesh speeds show the mesh being moved downwards to the wall to cluster in the boundary layer. The adaption is quite significant for a (11x11) mesh.

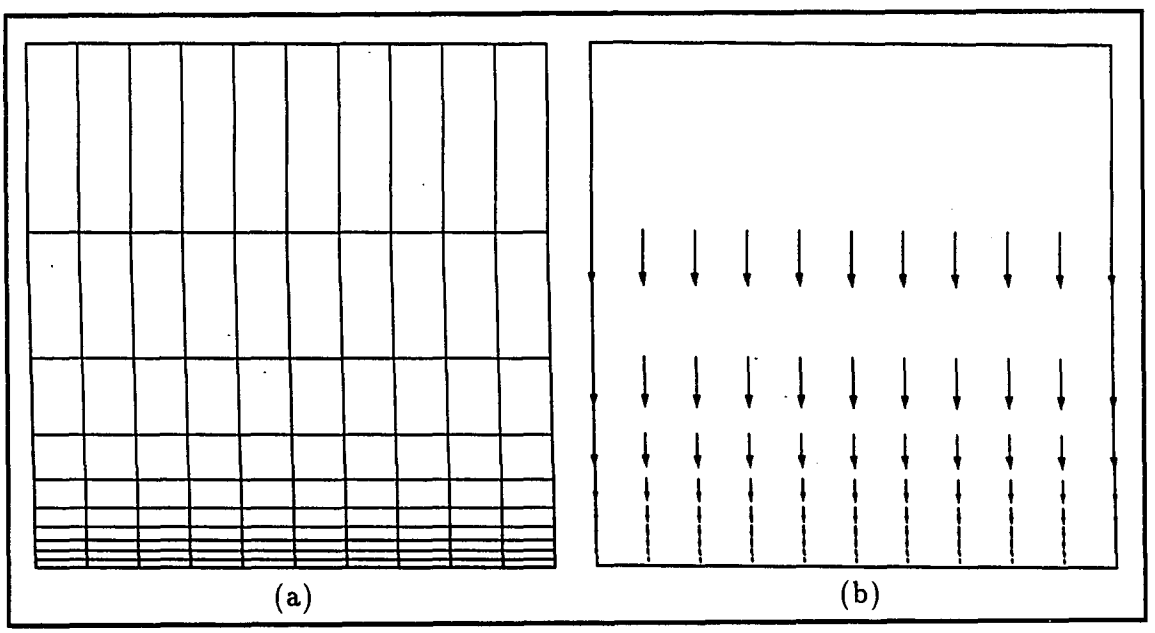

Figure 9.15: The (a) mesh and (b) mesh speeds at $t = 1.0$ seconds for the unit square with a quadratic boundary layer (case DS5)

CHAPTER 10. THE RESULTS FOR THE NAVIER-STOKES EQUATIONS

This chapter presents results from the application of the dynamically adaptive mesh method to the solution of the two-dimensional, unsteady Euler and Navier-Stokes equations. These results include inviscid flows through a converging-diverging nozzle, viscous flows over flat plates, and viscous, turbulent flows through a transonic diffuser. Each computation was performed on a static mesh so as to provide a baseline solution. The dynamic mesh computations were performed using the dynamically adaptive mesh method based on the computation of the mesh speeds using a backwards time-difference of the mesh, as well as the method based on the computation of the mesh speeds using the mesh speed equations. The results from the method using the mesh speed equations are presented for the inviscid flows in the converging-diverging nozzle and the viscous, subsonic flow over the flat plate. Comparisons to the results obtained from using a backwards time difference of the mesh are presented. For the remaining viscous flows, the dynamically adaptive mesh method uses the time-differenced mesh speeds. All the computations use the explicit, two-stage Lax-Wendroff method and the second-order Roe flux-difference splitting flux formula with Roe's Hyperbee limiter.

The Inviscid Flow Through a Converging-Diverging (CD) Nozzle

The flow through a converging-diverging (CD) nozzle is perhaps one of the most fundamental internal flows. When the quasi-one-dimensional assumptions can be applied and the flow is assumed to be steady, inviscid, and non-heat-conducting, one can obtain an exact solution for the flow in the nozzle. This is the classical result for a duct with varying area. The type of flowfield in the nozzle is determined by the ratio of the exit static pressure to the inlet static pressure. The flow is steady if the pressure ratio remains constant, but can be made unsteady if the pressure ratio varies in time. As the pressure ratio is lowered from a value of unity, the flow is accelerated through the duct. Eventually, a sonic velocity is reached at the throat and the flow becomes choked. For lower pressure ratios a shock exists in the diverging portion of the nozzle. This flow also is a model for a supercritical inlet for a high-speed aircraft. The shock serves to compress the flow prior to the engine intake face. The flows examined in this work mainly involve conditions in which a shock exists in the diverging portion of the nozzle or inlet. The flows are all unsteady; however, some reach a steady state at the end of the computation. Therefore, the quasi-one-dimensional theory can be used in some cases to provide an exact result for comparison. The unsteady computations do not have experimental or theoretical results that can be used for comparison. Data for comparison are obtained from a quasi-one-dimensional numerical computation. The numerical methods used for the quasi-one-dimensional computations are the same as used for the two-dimensional Euler equations; however, the methods are applied to include the quasi-one-dimensional assumptions. The quasi-one-dimensional computations use 290 evenly spaced mesh points to ensure a high level of accuracy.

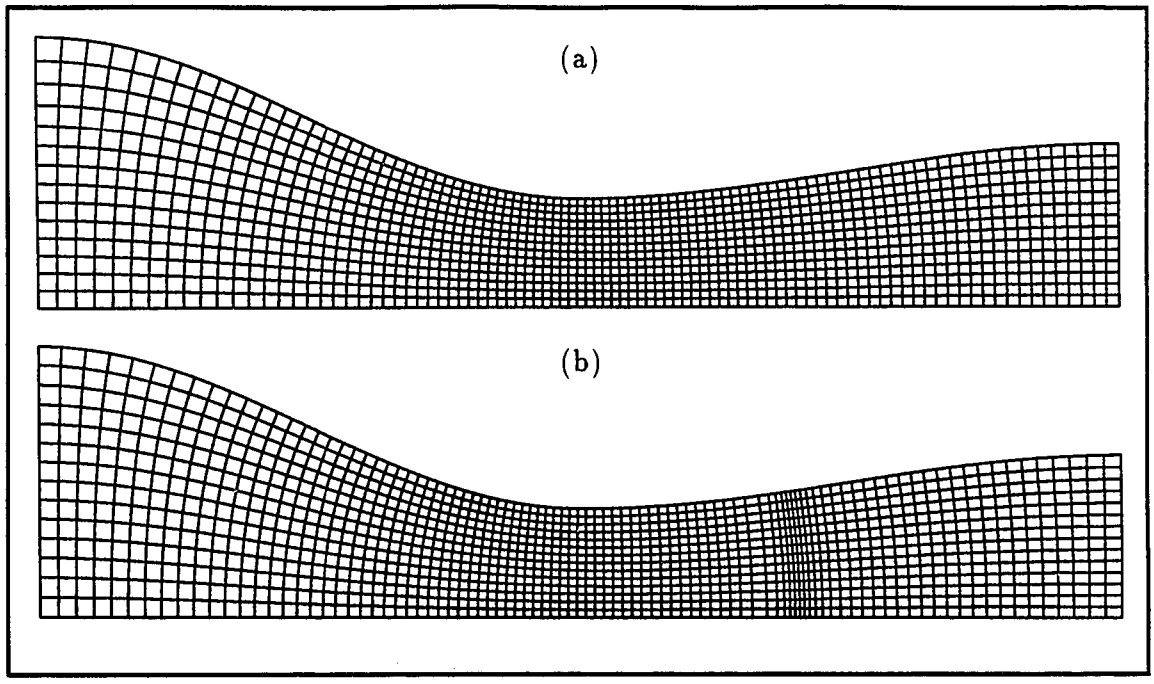

Figure 10.1: The geometry and mesh for the CD nozzle: (a) static mesh; (b) dynamically adapted mesh at the final time for the startup problem (case 0)

The geometry of the CD nozzle examined in this work is taken from reference [35]. The cross-sectional area is given as

$$S(x) = \begin{cases} 1.75 - 0.75 \cos\left[(0.2x - 1)\pi\right], & \text{if } 0.0 \leq x \leq 5.0 \\ 1.25 - 0.25 \cos\left[(0.2x - 1)\pi\right], & \text{if } 5.0 \leq x \leq 10.0 \end{cases} . \tag{10.1}$$

For the two-dimensional geometry, the bottom boundary was taken as a flat surface and the upper boundary was defined such that the height resulted in the proper $S(x)$ distribution. Figure 10.1(a) shows the two-dimensional geometry along with the mesh used for the static mesh computations.

For the proper comparison of the quasi-one-dimensional and two-dimensional solutions, the properties and time scales are non-dimensionalized. The time is nondi-

Table 10.1: Unsteady flow cases for the converging-diverging nozzle

Case	Description
0	Startup from total conditions
1A	10.0 % Step
1B	- 10.0 % Step
2A	20.0 % Impulse
3A	20.0 % Sinusoidal, 300 Hz
3B	10.0 % Sinusoidal, 50 Hz
3C	5.0 % Sinusoidal, 50 Hz

mensionalized by a residence time defined for the nozzle to be

$$t_{res} = L \; / \; V_{inflow}. \tag{10.2}$$

The L is the length of the nozzle, which is $L = 10$ units. The V_{inflow} is the fluid velocity at the inflow boundary. The t_{res} is the time required for a fluid particle to flow from the inflow boundary to the outflow boundary if it were travelling at the speed of V_{inflow}. For the quasi-one-dimensional problem, $V_{inflow} = 0.2395428$ units/sec and the residence time is 41.746193 seconds. For the two-dimensional CD nozzle problem, $V_{inflow} = 75.920331$ meters/sec and the residence time is 0.1317170 seconds.

The inviscid, converging-diverging nozzle flow can be made unsteady if the exit pressure is varied with time. The unsteady flow cases computed in this work are listed in Table 10.1.

The next four sections discuss each of these cases and the computational results. Comparisons are made between the computations from the quasi-one-dimensional equations and computations from the two-dimensional Euler equations on static and dynamic meshes.

Case 0: startup from total conditions

Case 0 starts with a stationary flow which is accelerated through the nozzle or inlet by a decrease in the exit pressure. The initial solution is uniform with the total pressure value of $p_t = 104074.6\ N/m^2$ and the total temperature value of $T_t = 252.9\ K$ and the velocity components u and v are zero. The exit pressure is then decreased from the total pressure value to a value of $p_{exit} = 84000\ N/m^2$ over a time interval of 10% of the residence time. A cubic spline curve with zero end-slope conditions is used for the transition. After the transition time interval, the exit pressure is held fixed. The steady-state flow solution is then obtained in which a shock exists in the diverging portion of the nozzle at $x = 6.9088$ units. This case is analogous to the startup procedure of a jet aircraft inlet in which the increasing RPM of the jet engine decreases the pressure at the engine inlet face.

The first computation was performed on a static mesh with dimensions of (95x15). The streamwise dimension of 95 was chosen to provide for a relatively high amount of resolution without excessive computations. The transverse dimension of 15 was chosen to result in finite-volume cells of aspect ratio close to unity. The mesh was generated with the parameters $\lambda_S = 1.0$, $\lambda_O = 1.0$, $\lambda_A = 0.0$, $\omega_G = 1.9$, and $\omega_{GB} = 0.9$. As mentioned in the previous chapter, setting $\lambda_A = 0.0$ resulted in mesh points being clustered in the throat region to satisfy smoothness and orthogonality requirements. The mesh was computed for 5000 iterations and reached a final residual of $7.2754x10^{-7}$ after starting from a residual of $4.0472x10^{-3}$. The computation required 65.431 CPU seconds on the NASA-Lewis Cray XMP. Figure 10.1(a) shows the mesh.

The computation of the flow on the static mesh started from the initial, uniform

solution based on total flow conditions as described above. The CFL number was $\nu = 0.7$. The entropy fix variable was set to $\epsilon = 1.0$. This value was determined by performing several computations with an increasing value of ϵ until the expansion shock at the sonic point was removed. This value of ϵ was used for the remaining flow computations for the converging-diverging nozzle. The computation was performed for 24000 time steps and reached a residual of about $4.0x10^{-6}$ after reaching a maximum of $6.3x10^{-2}$. Figure 10.4 shows the convergence history. The convergence history indicated some large scale changes in the solution over the first 5000 iterations, but then began a linearly decreasing path which had some small scale noise which was probably due to the second-order nature of the flux and the TVD limiter. The computation required 1150 CPU seconds on the Cray XMP. Figure 10.2(a) shows the Mach number contours at the final time. Figure 10.3 shows the comparison of the static pressures along the bottom wall with those from the quasi-one-dimensional theory. The pressures are nondimensionalized by the inflow static pressure. The comparison is very good; however, the resolution of the shock is limited to the mesh spacing of the static mesh.

A computation was performed with a dynamically adaptive mesh with the mesh speeds computed using a backwards time-difference of the mesh. The mesh adapted to the density gradients. An initial mesh with dimensions of (95x15) was computed using the parameters $\lambda_S = 1.0$, $\lambda_O = 1.0$, $\lambda_A = 1.0$, $\omega_G = 1.9$, and $\omega_{GB} = 0.9$. The initial solution was the uniform flow based on the total conditions as described above. Thus the initial mesh satisfied the mesh equations and was consistent with the solution as required for the start of the dynamic mesh adaption procedure. The flow computation was performed with a CFL number of $\nu = 0.7$ and required 16570 time

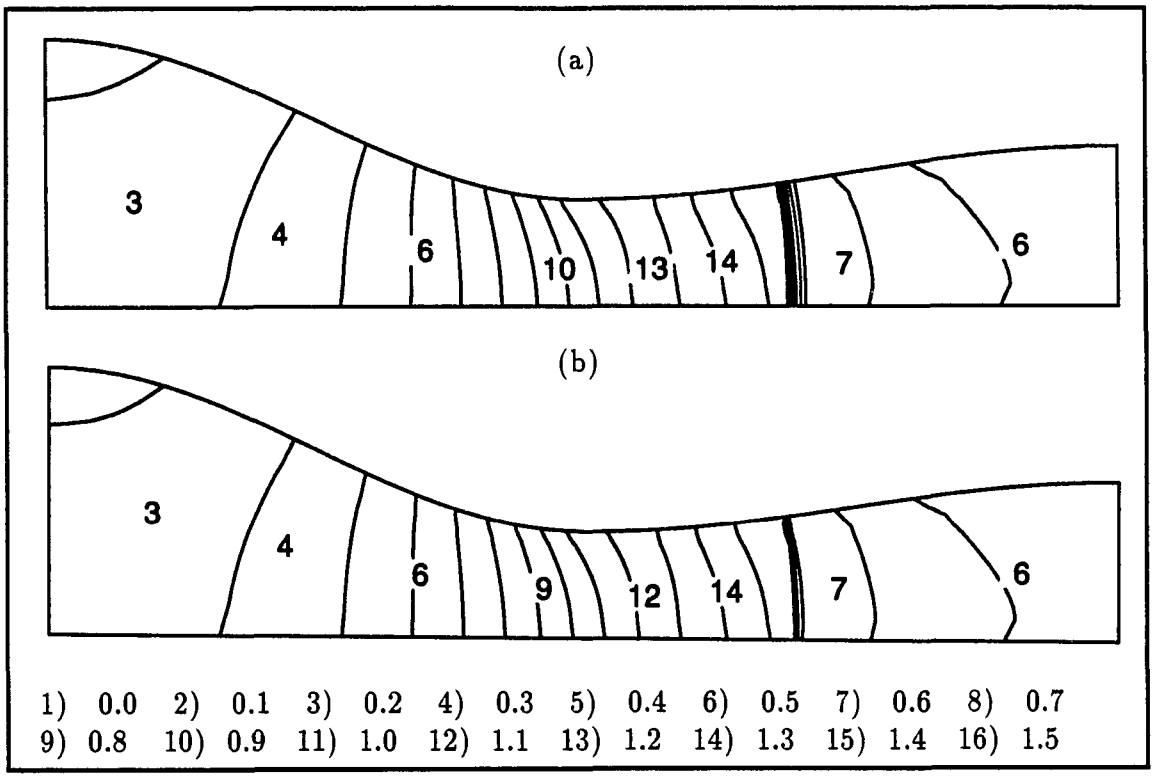

Figure 10.2: The Mach number contours for the steady-state solution for case 0: (a) static mesh; (b) dynamically adapted mesh with the mesh speeds computed from the mesh speed equations

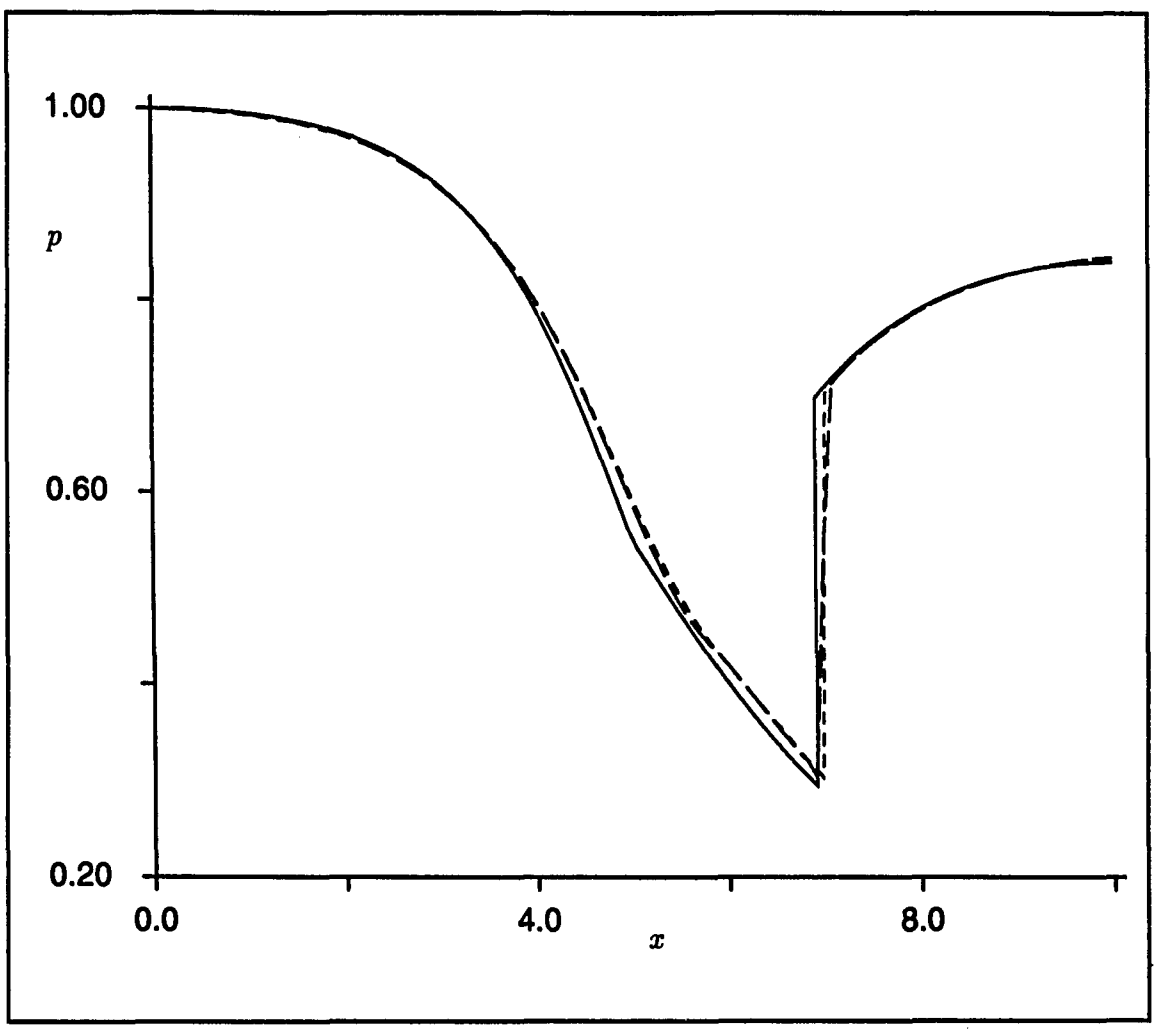

Figure 10.3: The pressure ratios along the bottom wall of the nozzle for the steady-state solution for case 0: (solid) quasi-one-dimensional theory; (dashed) static mesh; (dotted) dynamically adapted mesh using the mesh speed equations

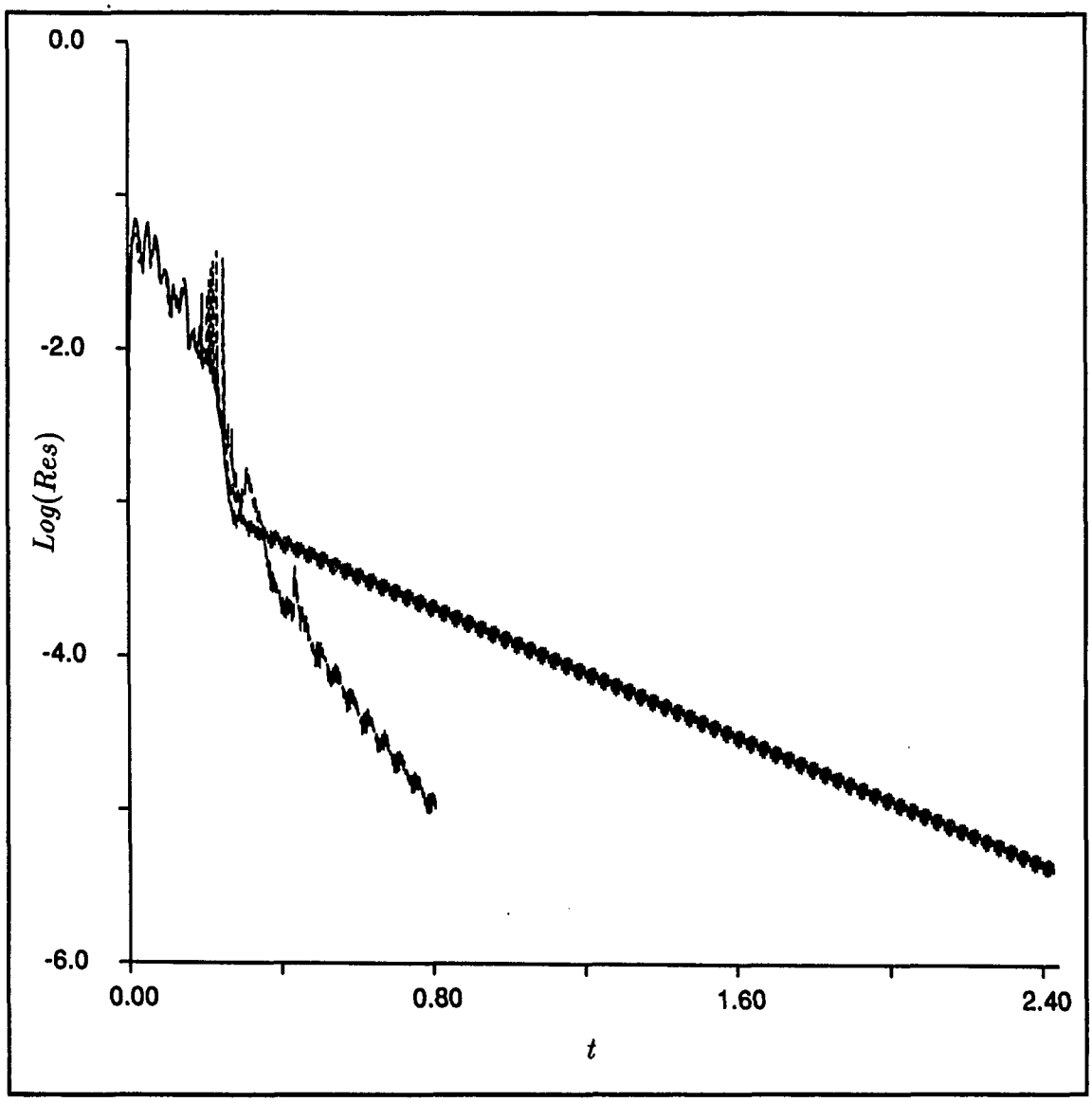

Figure 10.4: The convergence histories for the startup of the CD nozzle (case 0): (solid) static mesh; (dashed) dynamically adapted mesh using time-differenced mesh speeds; (dotted) dynamically adapted mesh using the mesh speed equations

steps and 1519 CPU seconds on the Cray XMP to reach a residual of $1.0x10^{-5}$. The mesh adaption parameters were $\lambda_A = 0.5$ and $\lambda_1 = 500$. Thirty smoothing passes were used to smooth the density solution prior to computing the weighting coefficient W and its derivatives. The mesh adaption performed one iteration of the mesh equations at each stage of the Lax-Wendroff two-stage method. The time-differencing of the mesh was then performed to obtain the mesh speeds. The dynamically adaptive mesh procedure calls for integrating the mesh speeds prior to solving the mesh equations. One would expect this to provide a better initial guess for the mesh iteration method. However, experience indicated that for this problem, performing the integration could cause the boundary of the mesh to deviate significantly from its true geometry. This in turn would introduce incorrect perturbations into the flow solution. Therefore, the mesh speeds were not integrated prior to solving the mesh equations for the computation computations using time-differenced mesh speeds. This seems reasonable since obtaining mesh speeds from backwards time difference does not enable the code to sense the behavior of the solution in the predictor stage. This could result in mesh speeds that when integrated, move the mesh in the direction other than the direction desired by the flow physics. It was also discovered that an instablity occurred when applying the inviscid wall boundary conditions for a dynamic mesh. The mesh points on the boundary are constrained to move along the boundary. Since the boundary conditions act only on information normal to the boundary, the mesh speeds do not influence the boundary conditions. However, the numerical implementation of the boundary conditions may cause some influence of mesh point motion. For the dynamic mesh computations, the mesh speeds were neglected in the numerical boundary condition computations.

A flow computation was performed with a dynamically adaptive mesh with the mesh speeds computed from the mesh speed equations. The initial mesh and flow solution were the same as used for the dynamically adaptive mesh computations with the time-differenced mesh speeds above. A CFL number of 0.6 was used and the time-integration was performed until a final time of $t = 0.3$ seconds. The computation required 4520 time steps and 987 CPU seconds on the Cray YMP. The adaption parameters were the same as for the computation using the time-differenced mesh speeds. The relaxation parameters were $\omega_{GS} = 1.0$ and $\omega_{GSB} = 1.0$. The mesh control law damping factor was set to a value of $\lambda_C = 50.0$, which was based on the results of the model problems. The iterations of the mesh speed equations were limited to 10 iterations per stage and the residual tolerance was set to $1.0x10^{-5}$. At the start of the computation, the mesh speed law required all 10 iterations. As the steady-state flow solution was reached, the residual fell below the tolerance and less iterations were required until only 1 iteration was required per stage. Figure 10.1(a) shows the adapted mesh. The stretching of the mesh just upstream of the shock appears to be an due to the smoothing of the gradient in density from the smoothing. Figure 10.2(b) shows the Mach number contours at the final time. The shock appears sharper than the contours for the static mesh result. Figure 10.3 shows the comparison of the pressures along the bottom wall with those from quasi-one-dimensional theory and the static mesh. The shock resolution is clearly improved.

The static mesh computation required $3.36x10^{-5}$ CPU seconds per time step per mesh point. The dynamically adaptive mesh computation using the time-differenced mesh speeds required $6.43x10^{-5}$ CPU seconds per time step per mesh point. This is a 91% increase in the amount of CPU effort. However one can see in Figure 10.4

that the convergence rate for the dynamically adaptive mesh is greater than for the static mesh. The static mesh required about 960 CPU seconds to converge to a residual of $1.0x10^{-5}$, while the dynamically adaptive mesh required 1516 CPU seconds. This makes the dynamically adaptive mesh method about 58% more expensive computationally. The dynamically adaptive mesh computation using the mesh speed equations required about $1.8532x10^{-4}$ CPU seconds per time step per mesh point on the Cray YMP. However, the YMP is faster than the XMP. It is estimated that the dynamically adaptive mesh method using the mesh speed equations requires about 10 times more computational effort than the method using a static mesh.

One important issue regarding dynamic meshes is whether there are any significant mesh-motion induced errors in the solution. Figure 10.5 shows the time history of the pressure at the location $x = 8.0$ units for the computations using a static mesh and the dynamic meshes. Figure 10.6 compares the pressures along the lower surface at the time $t = 0.1$ seconds. The time histories of the pressure and the solution from the static mesh and the dynamically adaptive mesh using the mesh speed equations coincide exactly. A small amount of error is seen for the dynamically adaptive mesh method using the time-differenced mesh speeds. From these two comparisons, it seems that the dynamically adaptive mesh method using the mesh speed equations result in less mesh-motion induced errors than the dynamically adaptive mesh method using the time-differenced mesh speeds.

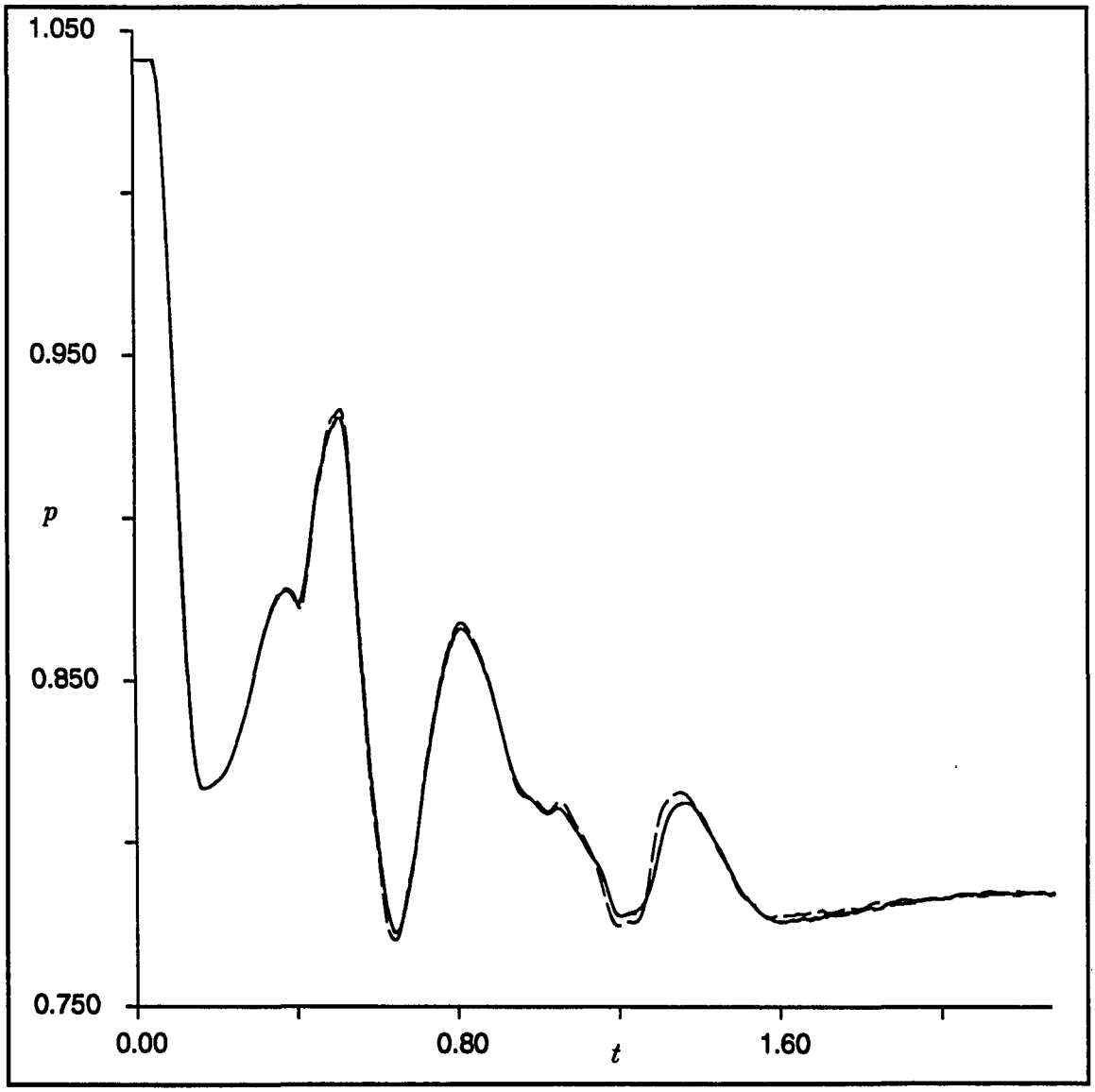

Figure 10.5: The time history of the pressure at the location $x = 8.0$ for the startup of the CD nozzle (case 0): (solid) static mesh; (dashed) dynamically adapted mesh using time-differenced mesh speeds; (dotted) dynamically adapted mesh using the mesh speed equations

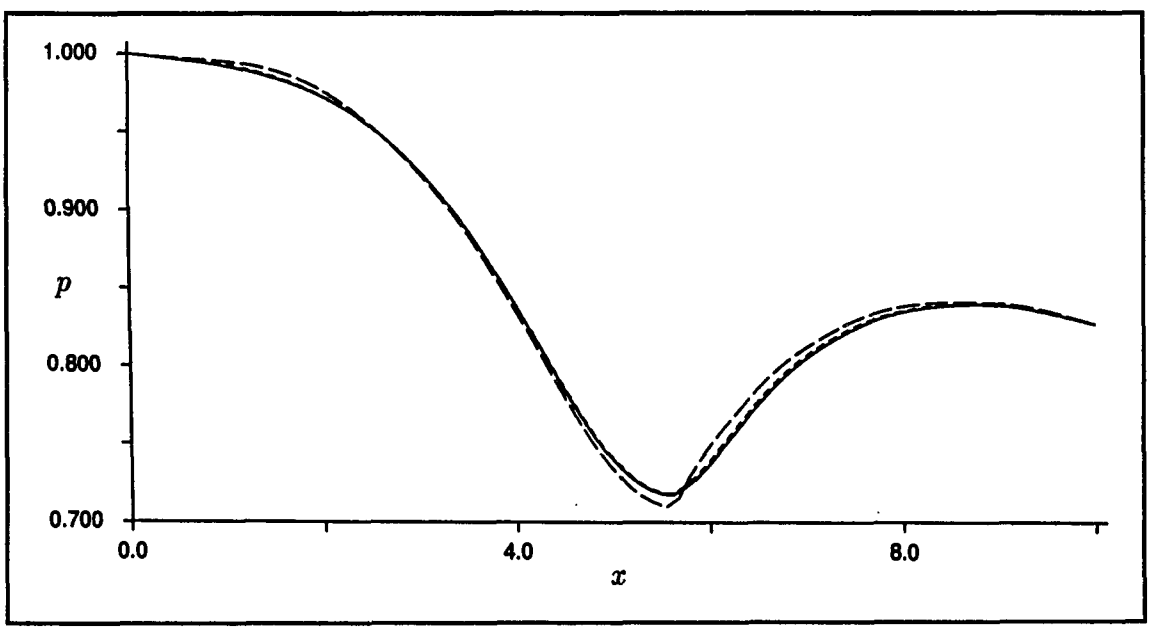

Figure 10.6: The solution at time $t = 1.0$ seconds for the startup of the CD nozzle (case 0): (solid) static mesh; (dashed) dynamically adapted mesh using time-differenced mesh speeds; (dotted) dynamically adapted mesh using the mesh speed equations

Case 1: a step variation of the exit pressure

Case 1 represents a step variation of the exit pressure. The initial flow and mesh solutions were the steady-state solutions obtained from case 0. The step variation in the exit pressure was specified by the amplitude of the pressure step expressed as a percentage of the inlet pressure. The transition in the step variation was taken over an interval of 1% of the residence time with a cubic spline curve defining the transition. The flow was unsteady while the new equilibrium was being formed. Case 1A involved a 10% step amplitude, while case 1B involved a -10% step amplitude. Case 1A forced the shock ahead of the throat to result in fully subsonic flow in the nozzle. Case 1B forced the shock downstream to a new shock location of $x = 8.007901$. The step variation of the exit pressure is analogous to a sudden change in the RPM of a jet engine, perhaps due to the throttling of the engine.

The computation for case 1A on the static mesh required 1780 time steps and 73 CPU seconds on the Cray XMP to reach a final time of $t = 0.16$ seconds. The dynamically adaptive mesh computation using the time-differenced mesh speeds required 2750 time steps and 257 CPU seconds on the Cray XMP with a CFL number of 0.6. The dynamically adaptive mesh computation using the mesh speeds computed from the mesh speed equations required 4040 time steps and 716 CPU seconds on the Cray YMP.

Figure 10.7 shows the Mach number contours at the final time for the dynamically adaptive mesh solution using the mesh speeds equations. Figure 10.8 shows the variation in the pressure at the location $x = 8.0$ units on the lower wall of the nozzle. There appears to be some type of high-frequency, low-amplitude disturbance passing the point after the main pressure wave has passed. One explanation is that these are

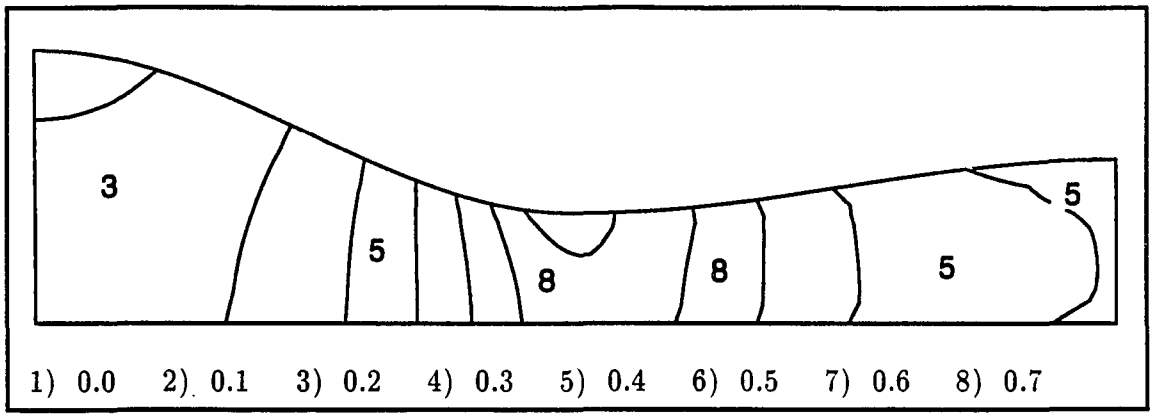

Figure 10.7: The Mach contours for case 1A at the final time for the dynamically adapted mesh solution using the mesh speed equations

pressure waves reflected from the shock. Another explanation is that they originate from the numerical method. Figure 10.9 shows the time history of the location of the shock. The shock is pushed upstream of the throat ($x = 5.0$ units) until the shock no longer exists and the flow becomes subsonic. The dynamically adaptive mesh computation using the mesh speeds equations seems to improve the computed shock locations over the shock locations computed using a static mesh as compared to the shock locations from the quasi-one-dimensional computations. Figure 10.10 compares the shock locations from the dynamic mesh computation using the mesh speed equations to the location of the minimum clustering of the mesh. This checks the performance of the shock tracking of the dynamic mesh. The tracking appears to be fairly good until the shock reduces strength as it approaches the throat.

The computations for case 1B on the static mesh required 1820 time steps and 78 CPU seconds on the Cray XMP to reach a time of $t = 0.16$ seconds. The computation on the dynamically adaptive mesh using the time-differenced mesh speeds required 3950 time steps and 366 CPU seconds on the Cray XMP with a CFL number of 0.6.

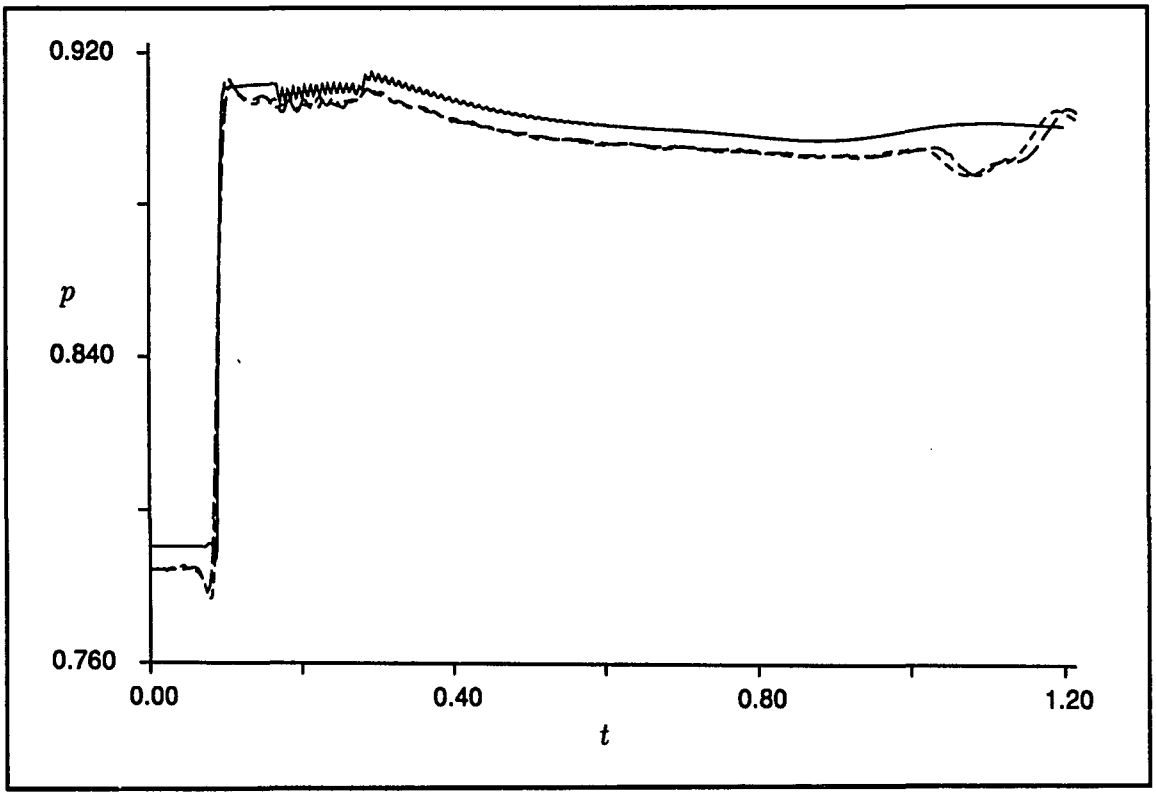

Figure 10.8: The time history of the pressure at location $x = 8.0$ units for case 1A: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

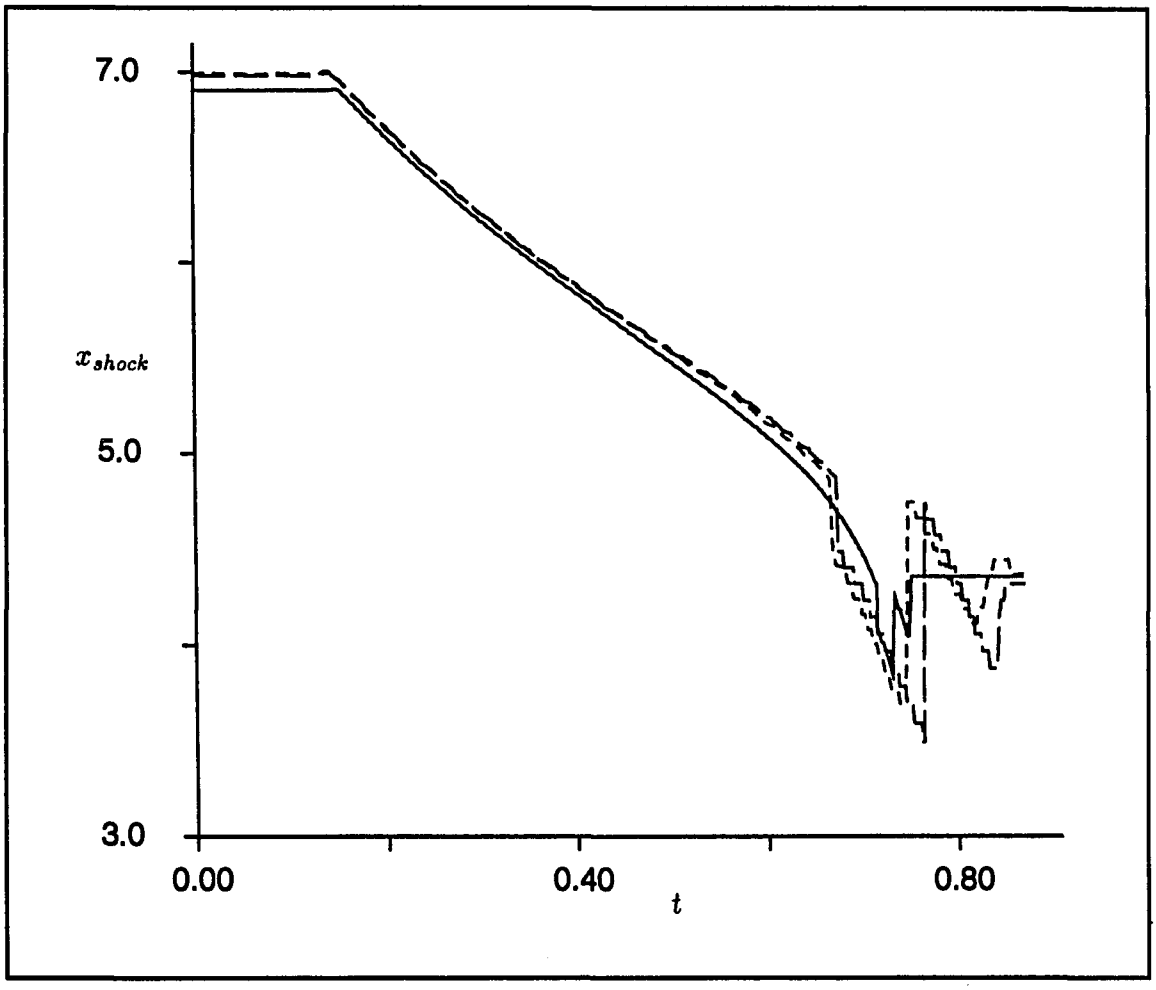

Figure 10.9: The time history of the location of the shock for case 1A: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

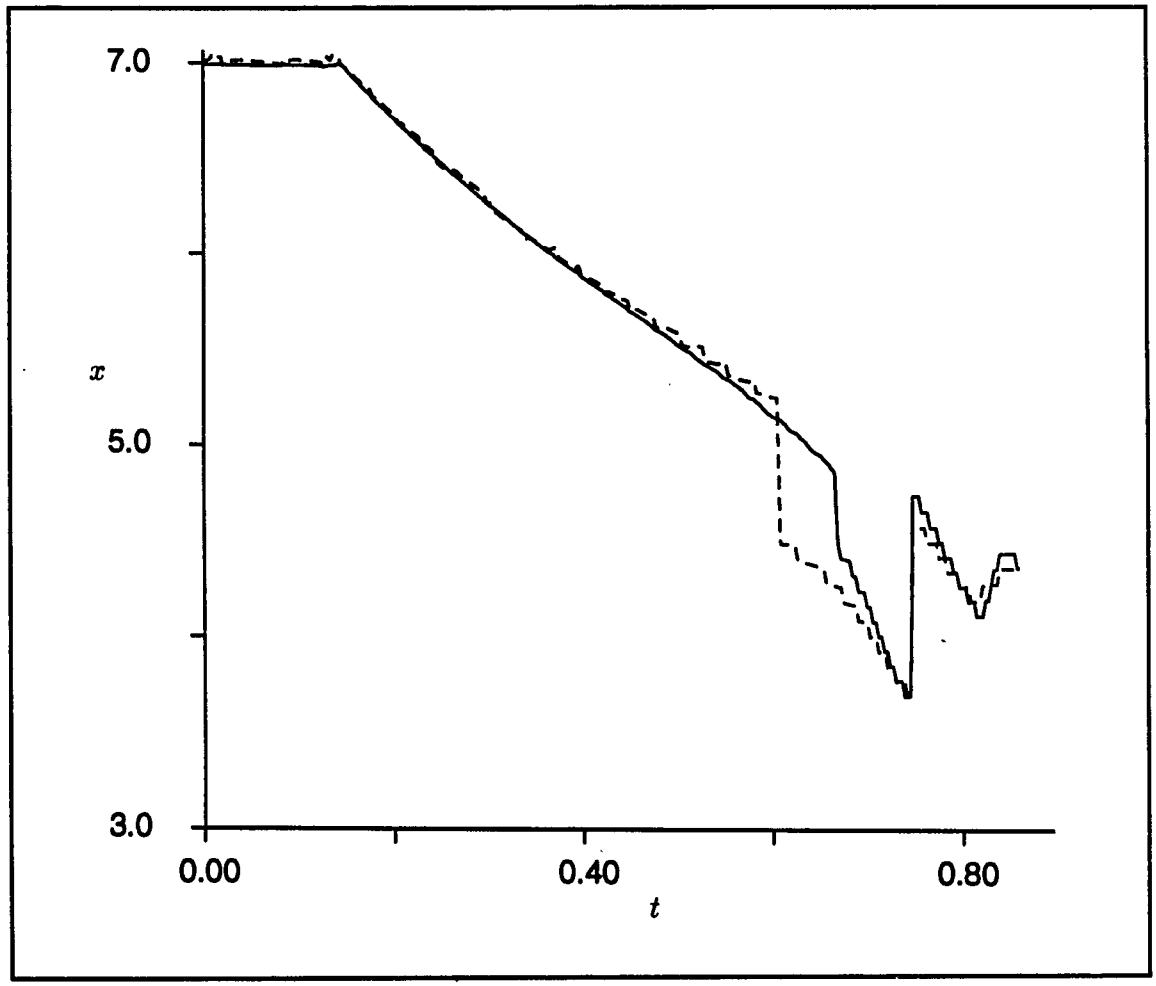

Figure 10.10: The comparison of the time history of the location of the shock for case 1A for the dynamically adapted mesh (solid) and the location of the minimum clustering (dashed)

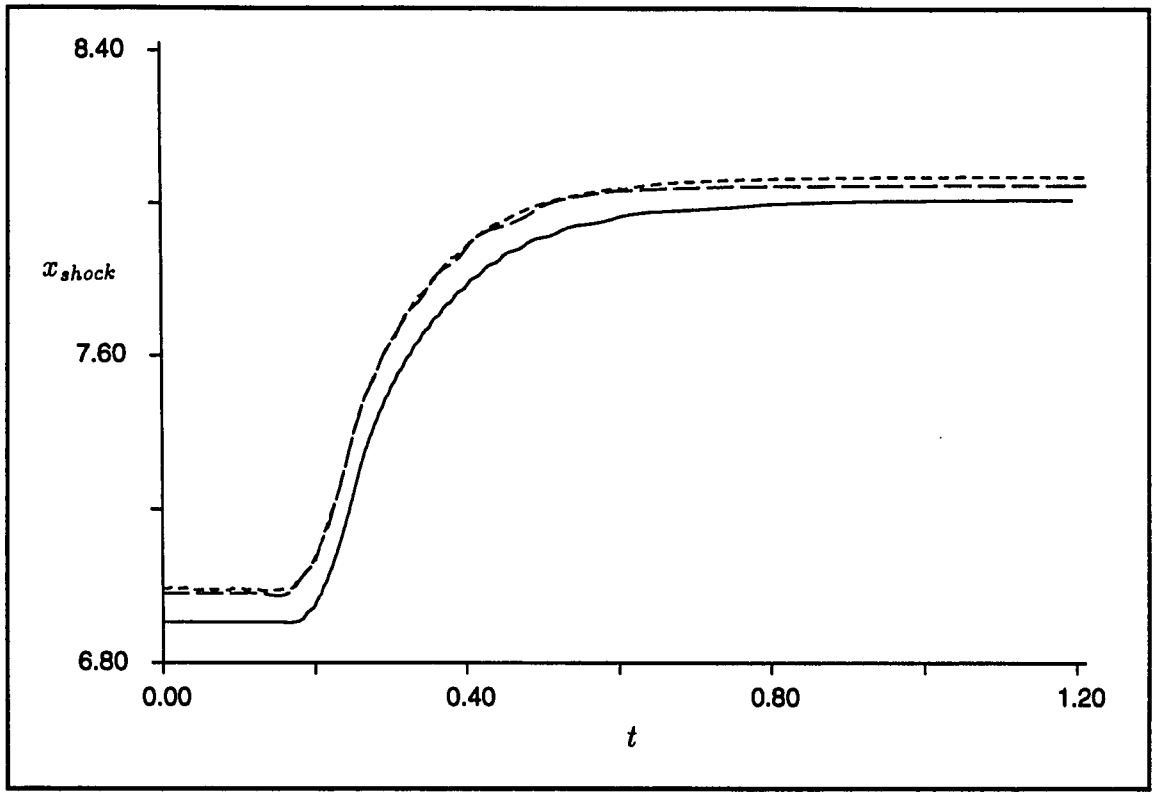

Figure 10.11: The time history of the location of the shock for case 1B: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

The computation on the dynamically adaptive mesh using the mesh speed equations required 4040 time steps and 495 CPU seconds on the Cray YMP. Figure 10.11 shows the time history of the shock location. Figure 10.12 examines the tracking of the shock by the mesh for the dynamically adaptive mesh method. The tracking seems to be rather good. The sharp changes in the position of the minimum clustering are due to the behavior of the numerical method for computing the location when mesh spacings are approximately equal at separate parts of the mesh.

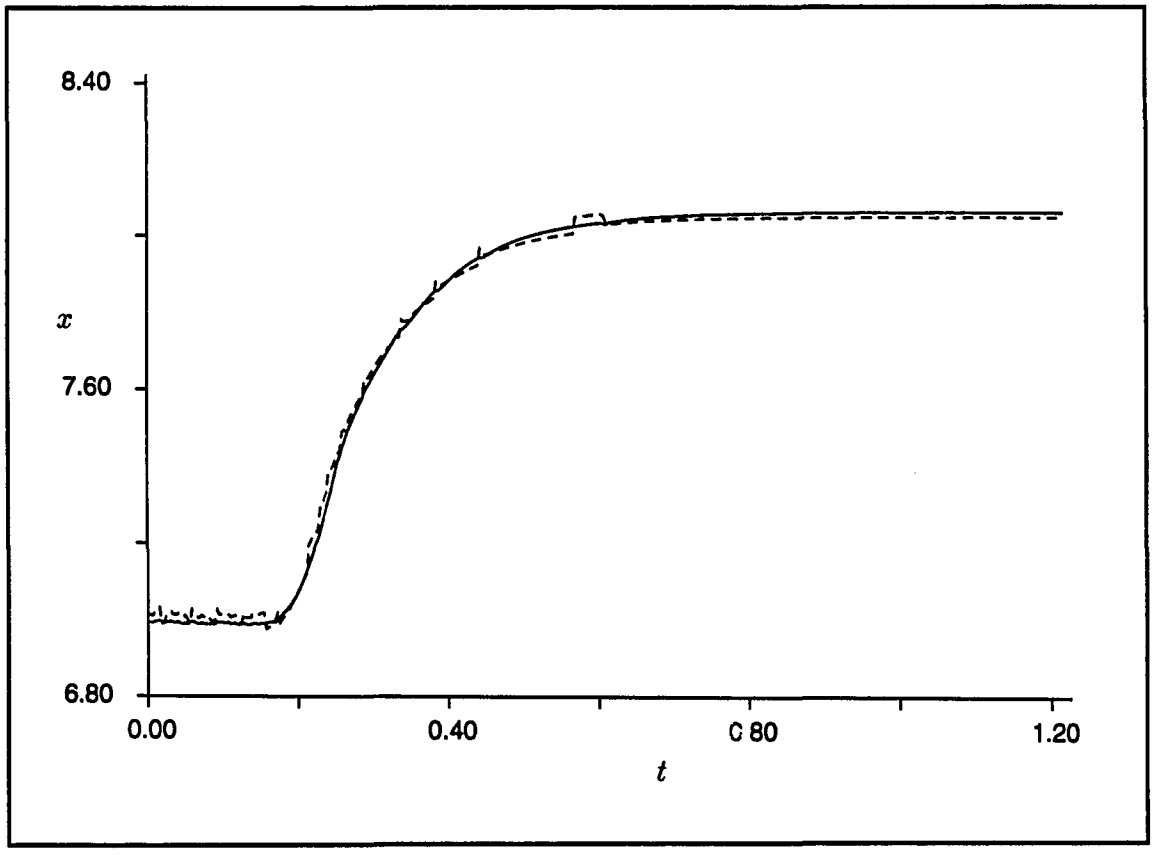

Figure 10.12: The comparison of the time history of the location of the shock for case 1B for the dynamically adapted mesh (solid) and the location of the minimum clustering (dashed)

Case 2: an impulse variation of the exit pressure

Case 2 represents an impulse variation of the exit pressure. The initial flow and mesh solutions are the steady-state solutions obtained from case 0. The impulse was specified as a pressure amplitude of 20% of the inlet static pressure. A cubic spline curve with zero-slope end conditions was used to transition the value of the pressure over a rise-time interval and a decrease-time interval. For the pulse duration, the value of the pressure was held fixed. The rise time, pulse duration, and decrease time for the case were 1%, 2%, and 1%, respectively, of the residence time. The impulse variation is analogous to a single pressure pulse that may evolve from a combustion instability in a scramjet engine. The computations were performed until a final time of $t = 0.16$ seconds.

The computation on the static mesh required 1820 time steps and 77 CPU seconds on the Cray XMP with the CFL number being 0.7. The computation on the dynamically adaptive mesh using time-differenced mesh speeds required 3740 time steps and 345 CPU seconds on the Cray XMP with the CFL number being 0.6. The computation on the dynamically adaptive mesh using the mesh speed equations required 3200 time steps and 403.939 CPU seconds on the Cray YMP with a CFL number of 0.7.

Figure 10.13 shows the time history of the pressure on the lower wall at the location $x = 8.0$ units. The pressure variation for the computations on the two-dimensional meshes compare well with the variation for the quasi-one-dimensional computation. This seems to indicate that the spatial resolution of the pressure impulse is adequate. Figure 10.14 shows the time history of the location of the shock along the lower wall. Figure 10.15 examines the shock tracking.

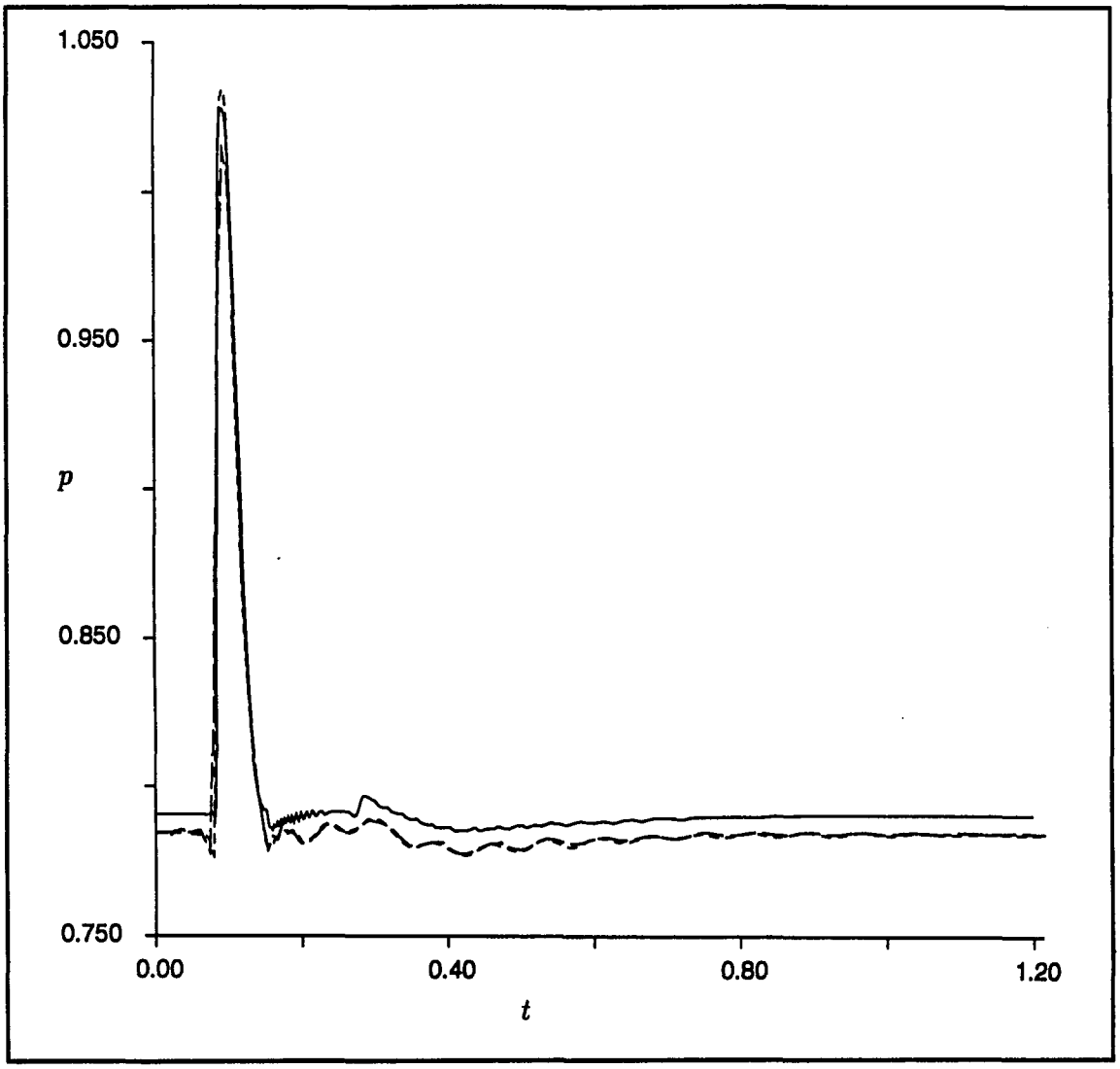

Figure 10.13: The time history of the pressure on the lower wall at location $x = 8.0$ units for case 2A: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

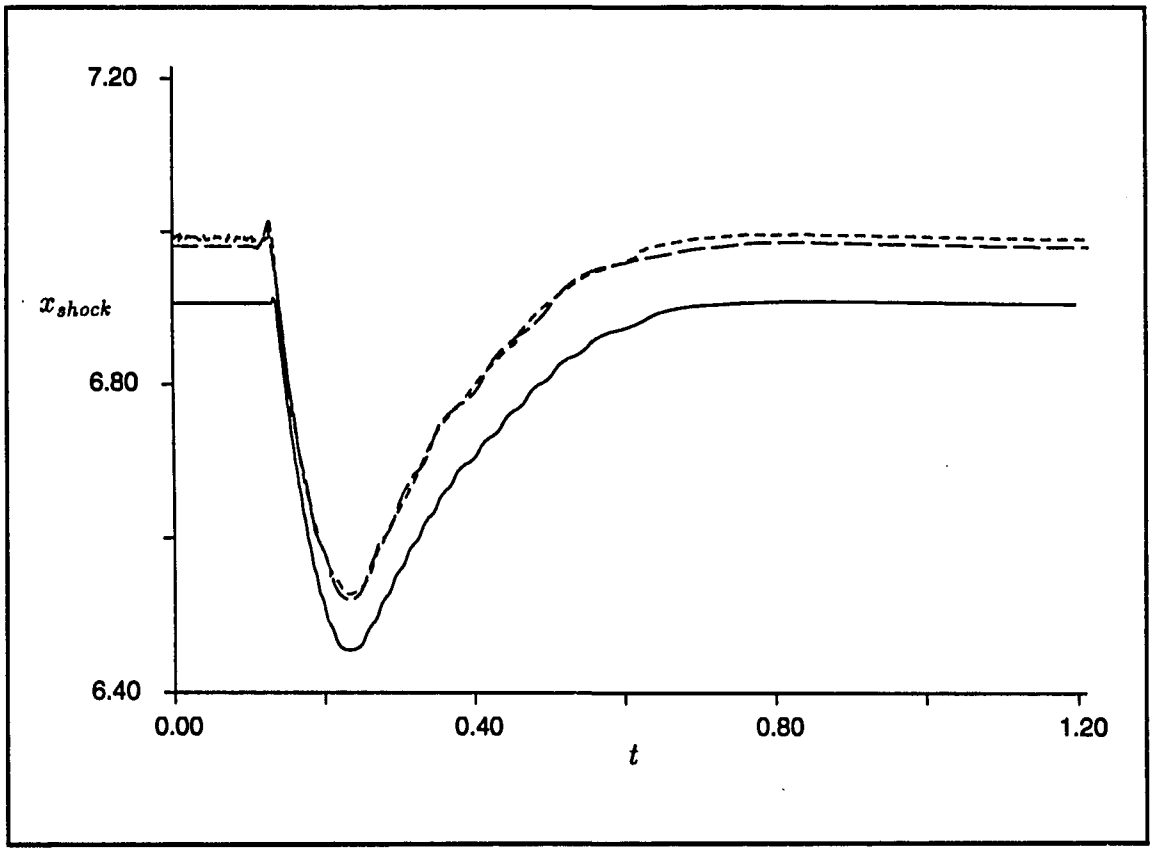

Figure 10.14: The time history of the location of the shock for case 2A: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

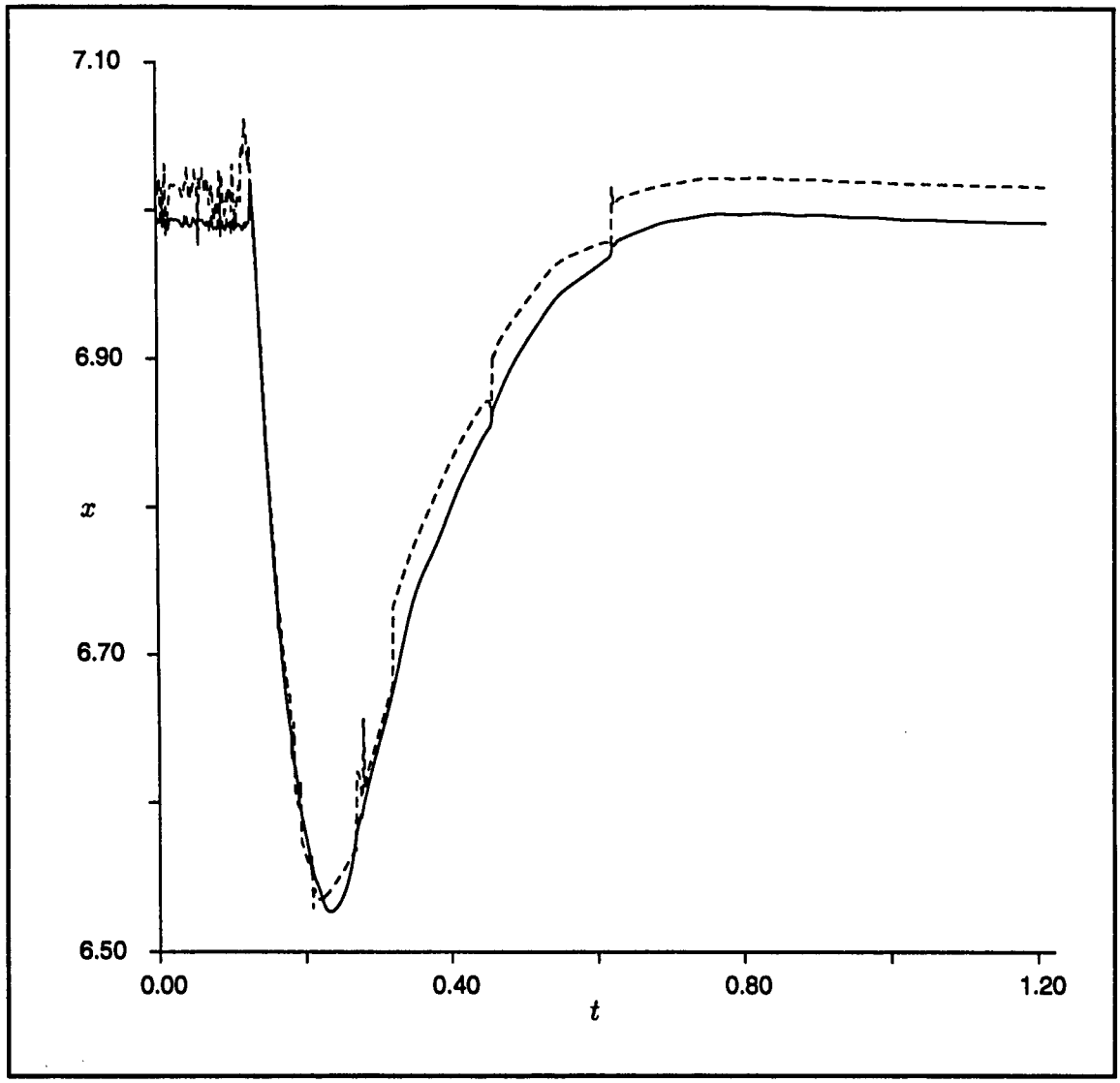

Figure 10.15: The comparison of the time history of the location of the shock for case 2A for the dynamically adapted mesh (solid) and the location of the minimum clustering (dashed)

Case 3: a sinusoidal variation of the exit pressure

Case 3 represents a sinusoidal variation of the exit pressure expressed as

$$P(t)_{exit} = P_{steady} + P_{amp} \sin \left[\omega_p \left(t - t_o \right) \right]. \tag{10.3}$$

The initial flow and mesh solutions are the steady-state solutions obtained from case 0, and so $P_{steady} = 84000 N/m^2$. The amplitude of the variation, P_{amp}, was specified as a percentage of the static pressure at the inlet. The P_{amp} was 20% for case 3A , 10% for case 3B, and 5% for case 3C. The frequency for case 3A was 300 Hz. The frequency for cases 3B and 3C was 50 Hz. The sinusoidal variation is analogous to a harmonic perturbation originating from an engine or combustion chamber.

The computation of case 3A on the static mesh required 2000 time steps and 80 CPU seconds on the Cray XMP to reach a final time of $t = 0.1$ seconds with a maximum time step of $5.0x10^{-05}$ and a CFL number of 0.7. The computation on the dynamically adaptive mesh using time-differenced mesh speeds required 2300 time steps and 212 CPU seconds on the Cray XMP with a CFL number of 0.6. The computation on the dynamically adaptive mesh using the mesh speed equations required 5000 time steps with a time step of $\Delta\tau = 2.0x10^{-5}$, and 948 CPU seconds on the Cray YMP. Figure 10.16 shows the time history of the pressure on the lower wall at the location $x = 8.0$ units. It appears that the meshes for the two-dimensional computations were not able to resolve the pressure variation spatially. Figure 10.17 shows the time history of the location of the shock along the lower wall. The shock moves towards the throat and then reaches a mean location with some small scale oscillation. Neither of the two-dimensional mesh solutions could pick up the small oscillation of the shock as shown by the quasi-one-dimensional results. This is probably

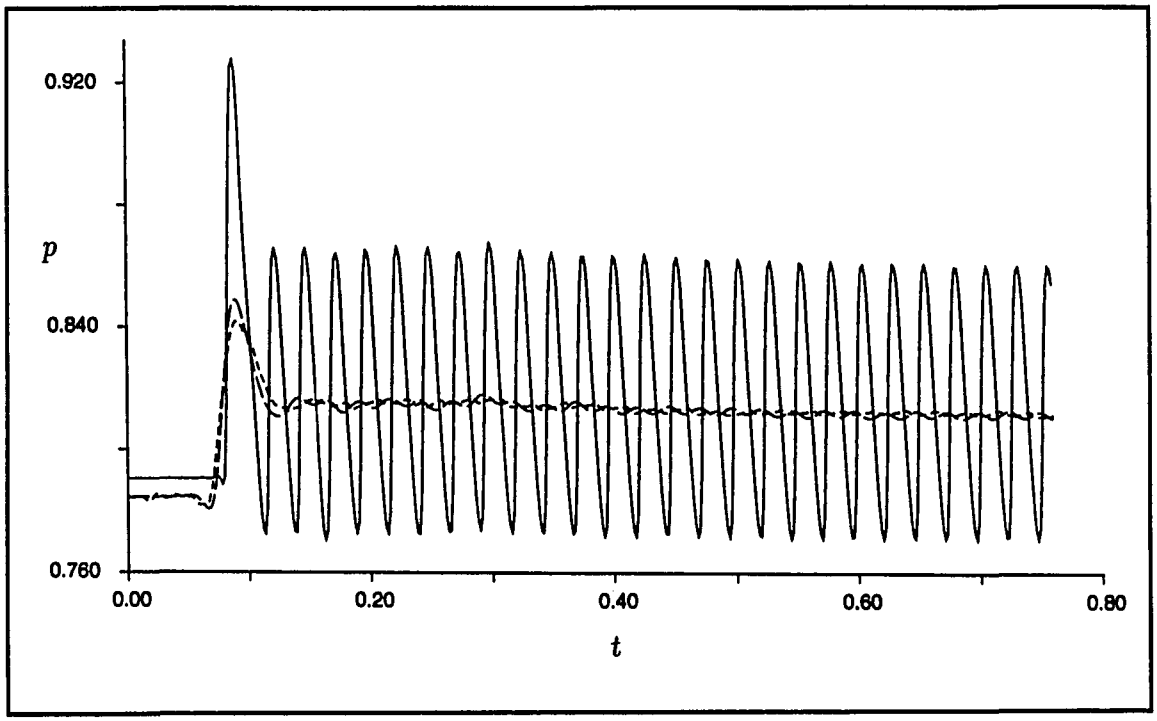

Figure 10.16: The time history of the pressure on the lower wall at the location $x = 8.0$ units for case 3A: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

due to poor spatial resolution of the pressure variation.

The computation of case 3B on the static mesh required 2280 time steps and 98 CPU seconds on the Cray XMP to reach a final time of $t = 0.20$ seconds with a maximum time step of $1.0x10^{-04}$ seconds and a CFL number of 0.6. The computation on the dynamically adaptive mesh using the time-differenced mesh speeds required 4190 time steps and 387 CPU seconds on the Cray XMP with a CFL number of 0.6. The computation on the dynamically adaptive mesh using the mesh speed equations required 3680 time steps with a CFL number of 0.7, and 700 CPU seconds on the Cray YMP. Figure 10.18 shows the time history of the pressure on the lower wall at

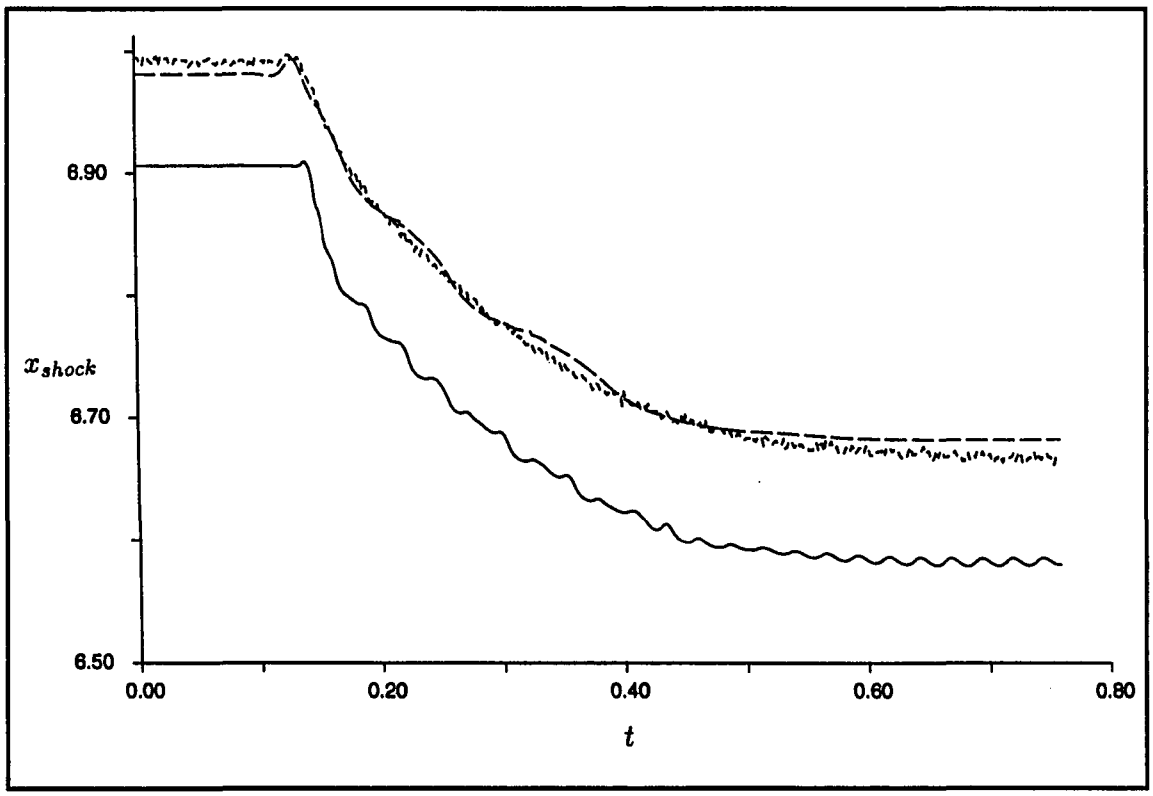

Figure 10.17: The time history of the location of the shock for case 3A: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

the location $x = 8.0$ units. There appears to be very little difference between the pressure history from the quasi-one-dimensional computations and the two-dimensional computations. This suggests that the spatial resolution is adequate to capture the pressure variation. Figure 10.19 shows the time history of the location of the shock along the lower wall. The shock moves towards the throat and then back towards the exit in response to the sinusoidal pulse. It appears then that the shock meets the next pulse that is moving towards the throat and they combine. The pulse moving towards the throat appears almost as a shock. The calculation of the shock location seems to have problems tracking two shocks and that may be the reason the shock location seems to be very sporadic. Figure 10.20 shows the pressure contours at the final time $t = 0.20$ seconds for the dynamically adaptive mesh computation. Figure 10.21 shows the pressures along the lower wall at the final time $t = 0.20$ seconds compared to the steady-state pressure. The two shocks are apparent.

The computation of case 3C on the static mesh required 2280 time steps and 98 CPU seconds on the Cray XMP to reach a final time of $t = 0.20$ seconds with a time step of $1.0x10^{-4}$ and a CFL number of 0.6. The computation on the dynamically adaptive mesh using time-differenced mesh speeds required 4580 time steps and 423 CPU seconds on the Cray XMP with a CFL number of 0.6. The computation on the dynamically adaptive mesh using the mesh speed equations required 3930 time steps with a CFL number of 0.7, and 749 CPU seconds on the Cray YMP. The time history of the pressure on the lower wall at the location $x = 8.0$ units showed that the pressure variation was well resolved at that point in the mesh. The pressure along the lower wall at the final time is shown in Figure 10.22. The form of the pressure variation is clearly seen. Figure 10.23 shows the time history of the location of the

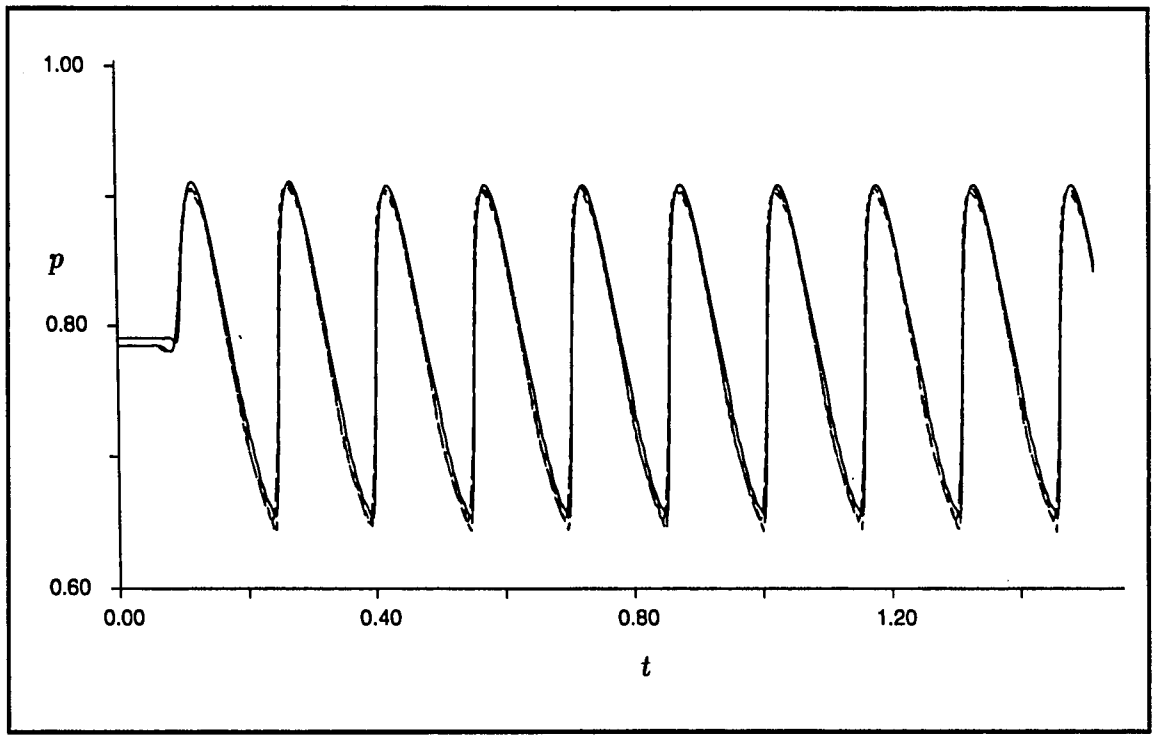

Figure 10.18: The time history of the pressure on the lower wall at the location $x = 8.0$ units for case 3B: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

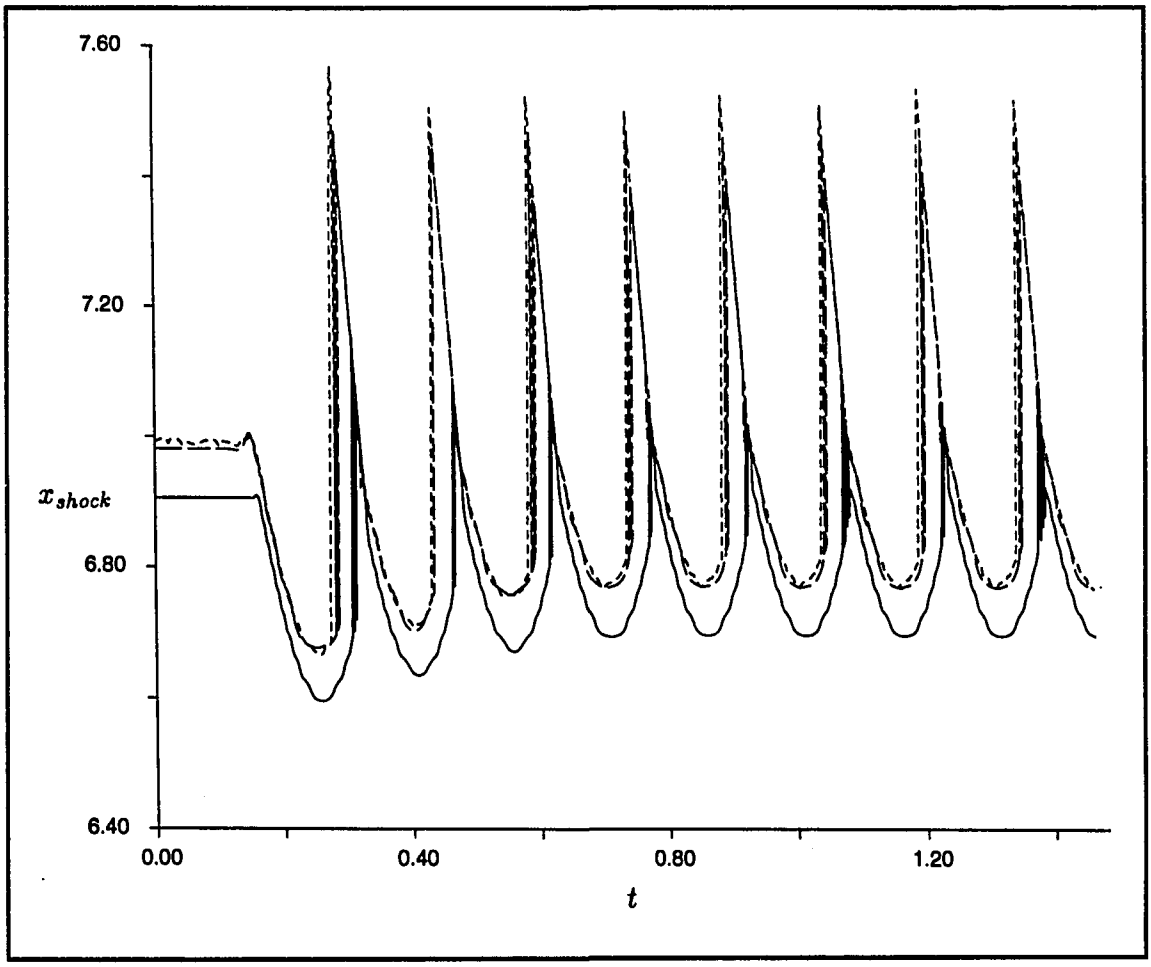

Figure 10.19: The time history of the location of the shock for case 3B: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

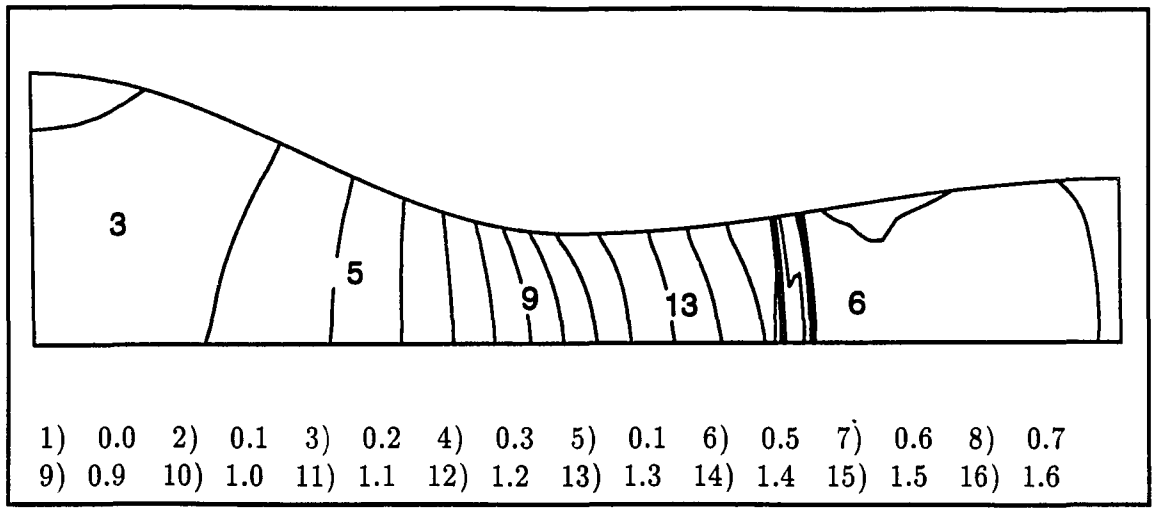

Figure 10.20: The Mach number contours for case 3B at the final time of $t = 0.20$ seconds for the dynamically adapted mesh

shock along the lower wall. The shock tracking is examined in Figure 10.24 in which the time history of the location of the shock is compared to the time history of the location of the point of minimum clustering in the mesh. The tracking of the shock does appear to lead the shock when the shock is moving down the nozzle and lag the shock when the shock is moving up the nozzle. Overall, the tracking apears good.

The computations involving the converging-diverging nozzle demonstrated that the dynamically adaptive mesh method was capable of improving the resolution of the shock tracking in inviscid flowfields. The method was not able to resolve the high-frequency oscillations of case 3A. The dynamically adaptive mesh method using the mesh speed equations was shown to be more accurate than the method using the time-differenced mesh speeds; however, the dynamically adaptive mesh method using the mesh speed equations was not shown to be more accurate than the method using static meshes.

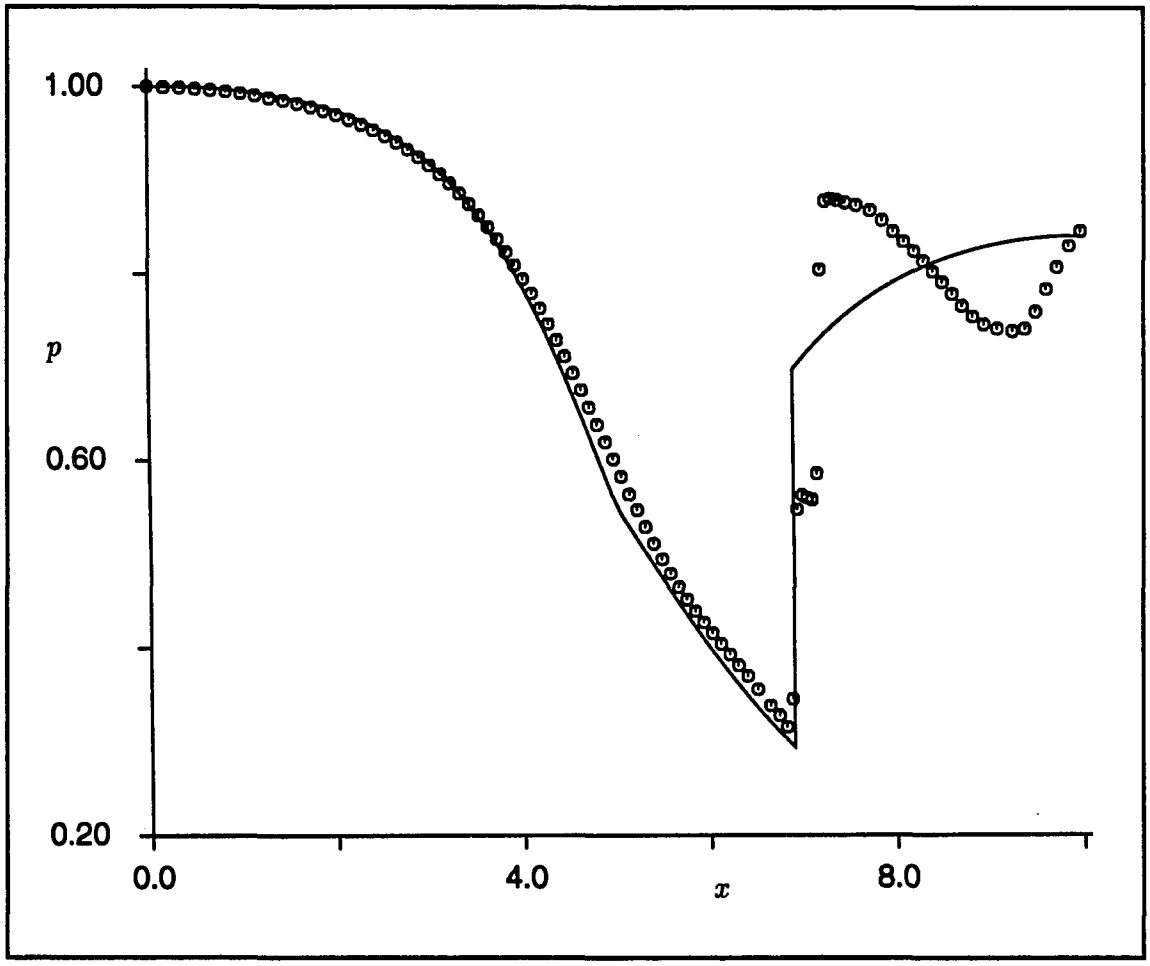

Figure 10.21: The pressure along the lower wall for case 3B at $t = 0.20$ seconds (circles) compared to the steady-state solution (solid)

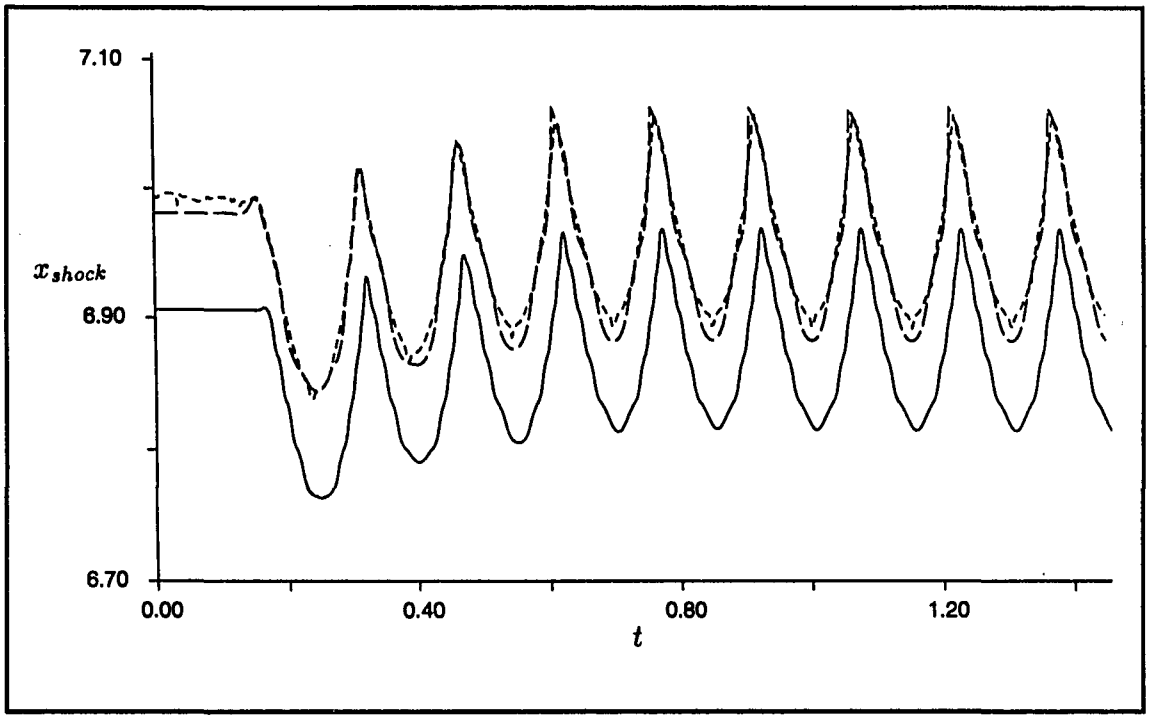

Figure 10.22: The time history of the location of the shock for case 3C: (solid) quasi-one-dimensional; (dashed) static mesh; (dotted) dynamically adapted mesh

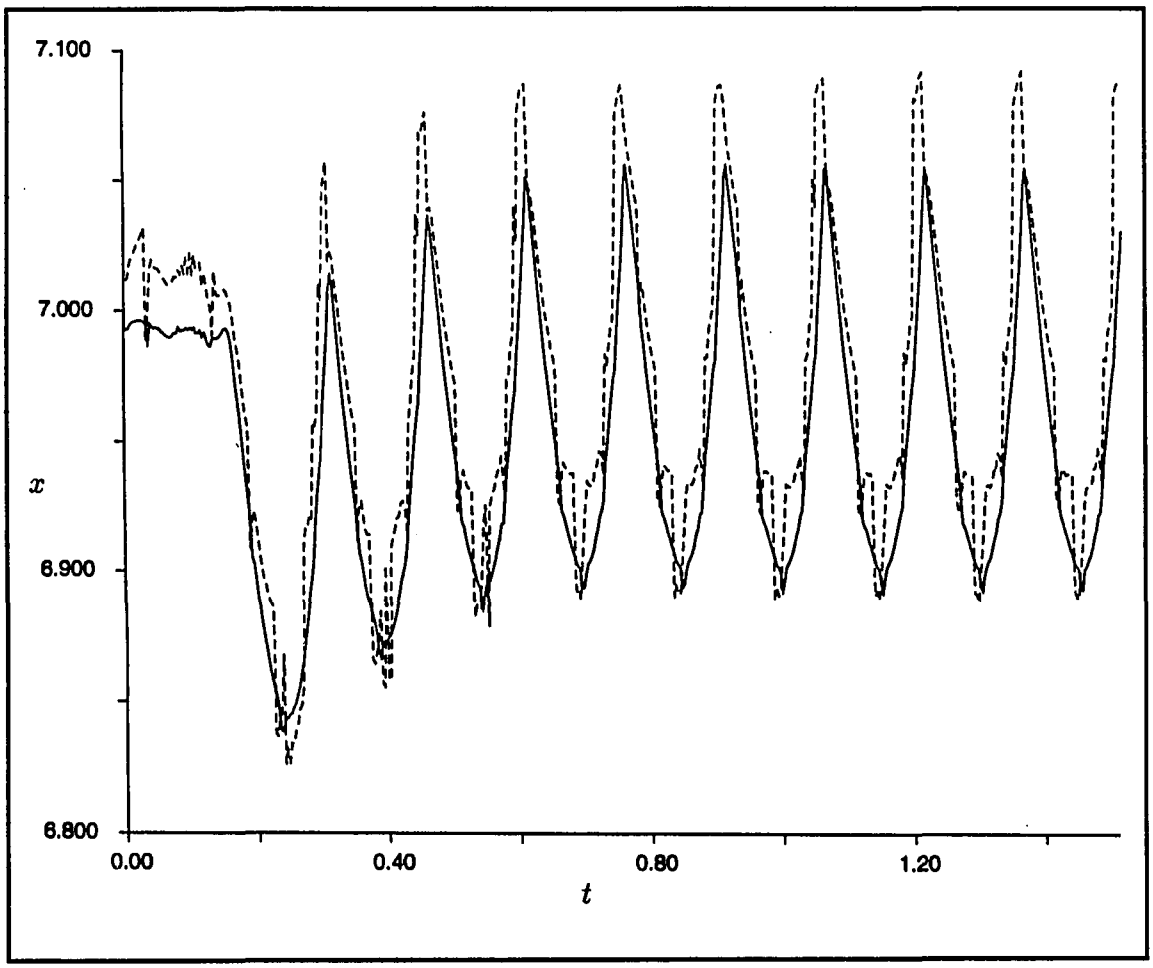

Figure 10.23: The time history of the location of the shock for case 3C for the dynamically adapted mesh (solid) compared to the location of the minimum clustering (dashed)

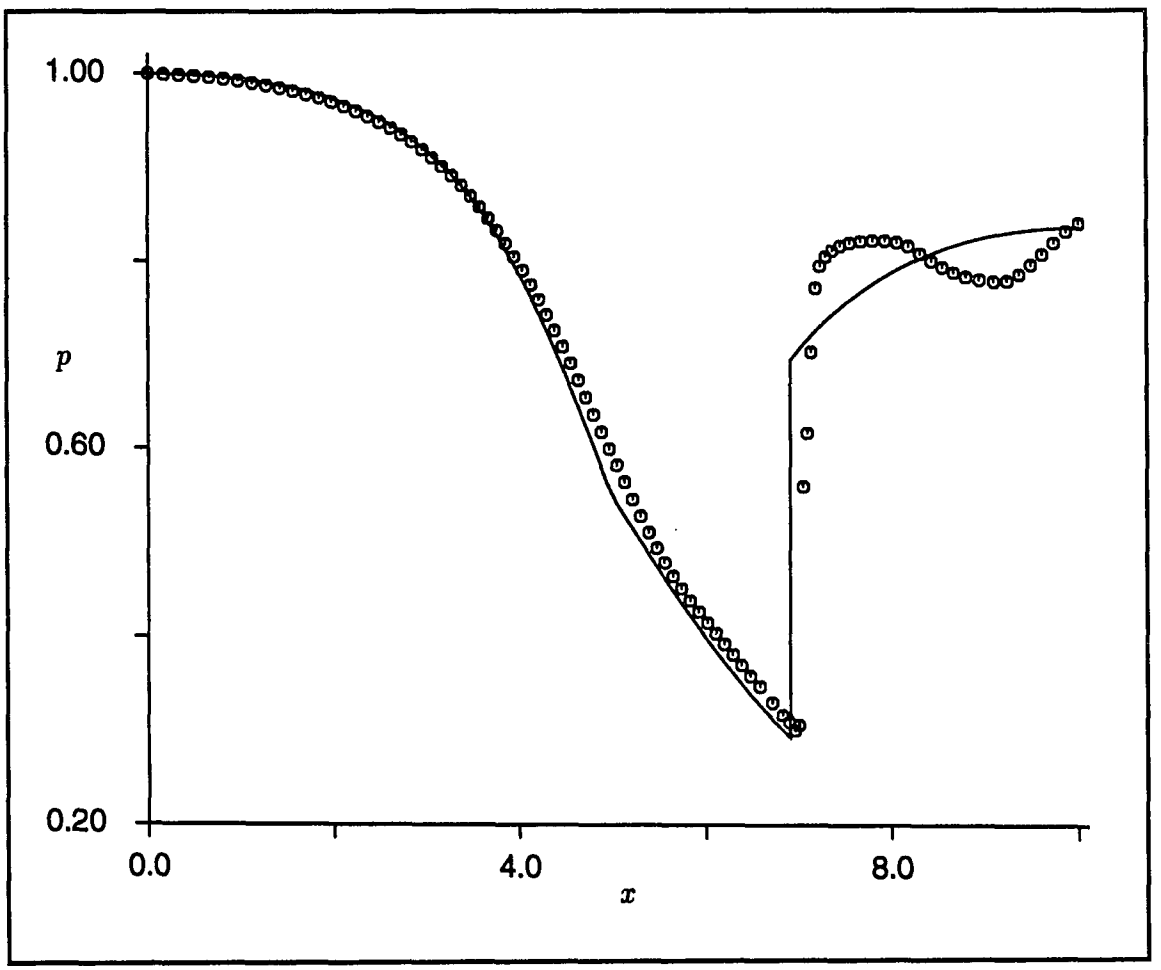

Figure 10.24: The pressure along the lower wall for case 3C at $t = 0.20$ seconds (circles) compared to the steady-state solution (solid)

The Subsonic, Viscous Flow over a Flat Plate

The application of the dynamically adaptive mesh method to viscous flows is first demonstrated through the computation of the subsonic, viscous flow over a flat plate. The exact solution for the incompressible, laminar boundary-layer equations can be computed from the Blasius boundary-layer theory. The freestream flow conditions were a Mach number of $M = 0.5$, a Reynolds number of $Re = 1.0x10^5$ based on a reference length of $L = 1.0$ meters, and a temperature of $T = 200\ K$. The flow was also computed using a compressible boundary layer code.

The computational domain was a rectangle of length 2.0 meters, which was the length of the flat plate, and of height 1.0 meters. Subsonic inflow boundary conditions were applied on the left boundary. Subsonic outflow boundary conditions were applied on the top and right boundaries. The bottom boundary was the plate and viscous boundary conditions were applied with an adiabatic temperature boundary condition.

The computation on the static mesh was performed for 20000 time steps with a CFL number of 0.8, and required 1103 CPU seconds on the Cray XMP. The residual started at $4.76x10^{-1}$ and was reduced to $2.4x10^{-4}$. A (41x41) mesh was used in which the constant-η mesh lines were clustered near the plate using the Roberts stretching function [5] with a stretching parameter of 1.008. This put the first point at a distance of $8.06x10^{-4}$ from the plate. The constant-ξ lines were slightly clustered near the leading edge.

The computation on the dynamically adaptive mesh using time-differenced mesh speeds was performed for 20000 time steps with the CFL number being 0.7, and required 2081 CPU seconds on the Cray XMP. The residual started at $4.6x10^{-2}$

and was reduced to a value of $2.5x10^{-6}$. The computation started with freestream conditions and a uniformly spaced (41x41) mesh. Thus the mesh satisfied the mesh equation and was consistent with the solution. The tangential velocity boundary conditions at the plate were transitioned from the freestream value to an adherence value of zero using a cubic spline curve. The transition time was $1.411x10^{-3}$ seconds, which was 10% of the residence time. The mesh weighting function used in the mesh adaption was the gradient in (ξ,η) space of the Mach number. The adaption parameters were $\lambda_S = 1.0$, $\lambda_O = 2.0$, $\lambda_A = 2.0$, $\lambda_0 = 1.0$, $\lambda_1 = 5000.0$, and 30 smoothing passes were used. The mesh equations were iterated once for the predictor stage and once for the corrector stage. The mesh adapted quickly to the developing boundary layer. There was very little change in the mesh over the last two-thirds of the computation. Therefore, for steady flows, it would be more efficient if the mesh adaption was totally turned off after the flow was mostly developed. The adaption placed the first point from the plate at a distance of about $2.0x10^{-3}$ units.

The computation on the dynamically adaptive mesh using the mesh speed equations was performed for 11200 time steps with the CFL number being 0.8, and required 1618 CPU seconds on the Cray XMP. The adaption parameters were the same as for the computation on the dynamically adaptive mesh using time-differenced mesh speeds; however, $\lambda_O = 1.0$ and $\lambda_A = 3.0$. A mesh control law damping factor of $\lambda_C = 50.0$ was used. The flow residual was reduced to a value of $8.6x10^{-4}$. The iterations of the mesh speed equations were limited to 5 iterations per stage with a convergence tolerance of $1.0x10^{-5}$. At the start, all 5 iterations were needed; however, as the boundary layer began to form, only 1 iteration of the mesh speed equations was needed. The mesh spacing of the first mesh point was $1.729x10^{-3}$

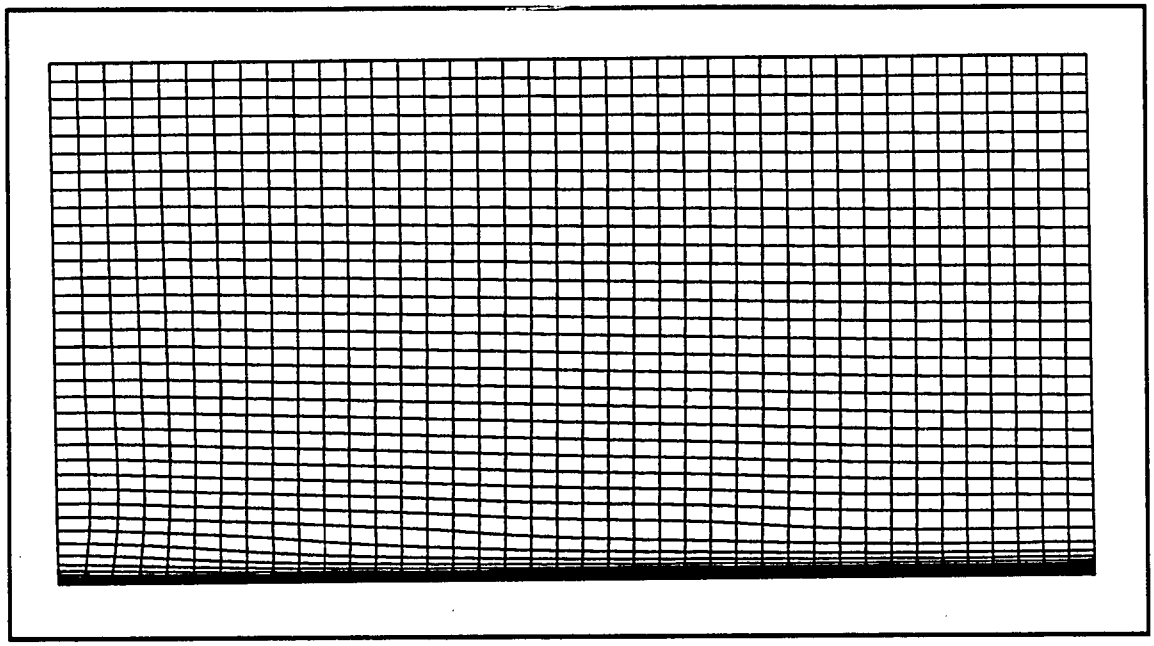

Figure 10.25: The final (41x41) dynamically adapted mesh for the subsonic boundary-layer computation using the mesh speed equations

units. Figure 10.25 shows the final mesh from the computation on the dynamically adaptive mesh. Figure 10.26 shows the u velocity profile at the location $x = 1.0$ meters. The comparison to the results from a compressible boundary layer code is very good. There is a slight overshoot near the edge of the boundary layer. This is most likely due to the Roe flux-difference splitting. Figure 10.27 shows the temperature profile. The computation on the dynamically adaptive mesh underpredicts the temperature at the plate. This error is most likely due to improper resolution of the mesh normal to the plate.

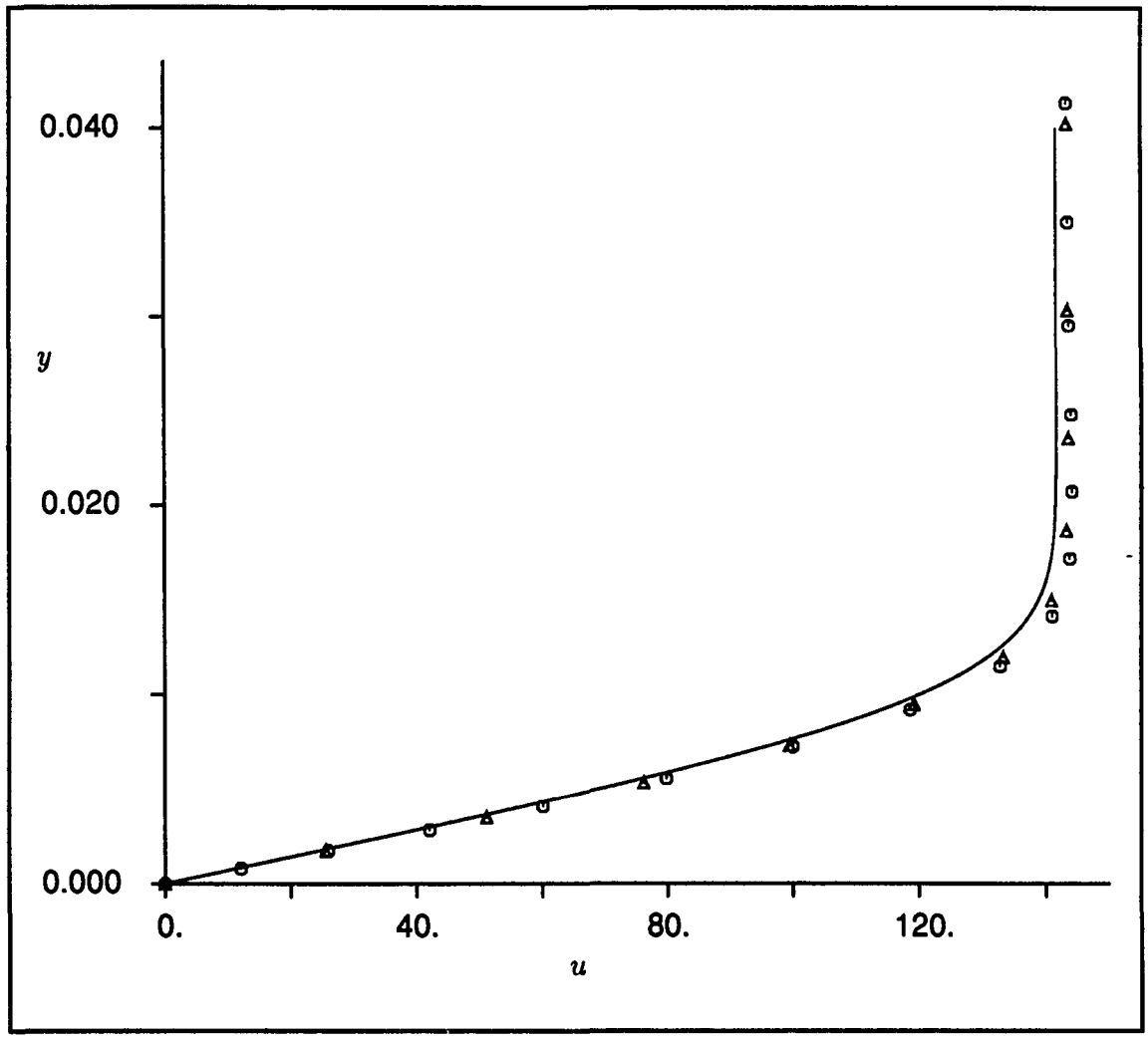

Figure 10.26: The u velocity profile at $x = 1.0$ units for the subsonic boundary-layer computation: (solid) boundary-layer code; (circles) static mesh; (triangles) dynamically adapted mesh

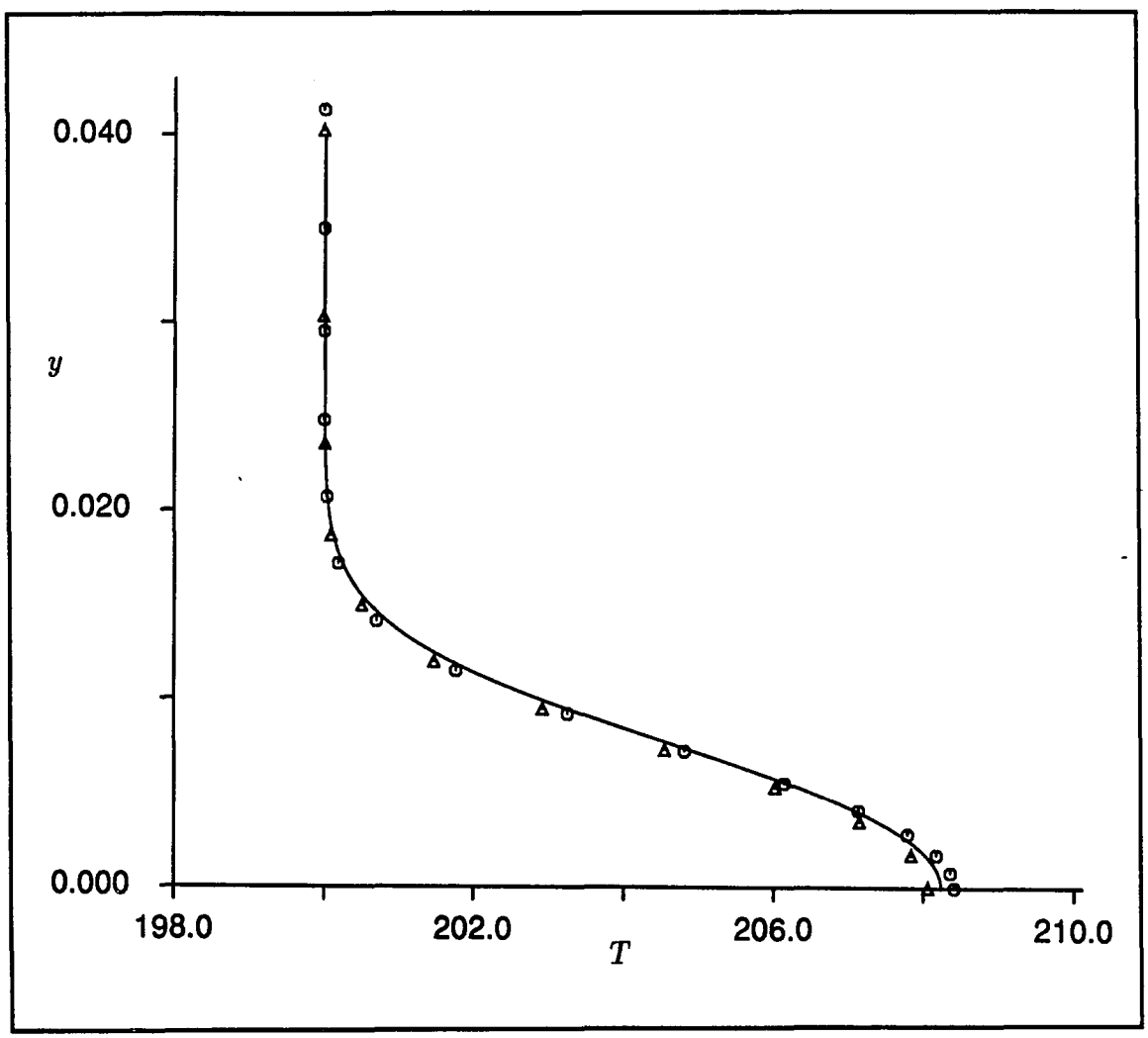

Figure 10.27: The temperature profile at $x = 1.0$ units for the subsonic boundary-layer computation: (solid) boundary-layer code; (circles) static mesh; (triangles) dynamically adapted mesh

The Supersonic, Viscous Flow over a Flat Plate

A further application of the dynamically adaptive mesh method to viscous flows is demonstrated through the computation of supersonic, viscous flow over a flat plate. The freestream flow conditions were a Mach number of 2.0, a Reynolds number of $Re = 1.65x10^6$ based on a reference length of $L = 1.0$ meters, and a temperature of $T = 221.6\ K$. The computed results can be compared to results from a compressible boundary-layer code and a parabolized, Navier-Stokes code.

The computational domain was a rectangle of length 2.0 meters, which was the length of the flat plate, and of height 0.5 meters. Supersonic inflow boundary conditions were applied on the left boundary. Subsonic outflow boundary conditions were applied on the top boundary. Supersonic outflow boundary conditions were applied on the right boundary. The bottom boundary was the plate, and viscous boundary conditions were applied with the temperature fixed at a value of 221.6 K.

A (31x41) static mesh was defined in which the constant-η mesh lines were clustered near the plate using the Roberts stretching function [5] with a stretching parameter of 1.0009. This put the first point at a distance of $1.067x10^{-4}$ from the plate. The computation on the static mesh was performed for 25000 time steps with the CFL number being 0.8, and required 1104 CPU seconds on the Cray XMP. The residual started at $6.0x10^1$ and was reduced to $8.3x10^{-3}$.

The computation on the dynamically adaptive mesh using time-differenced mes speeds was performed for 20000 time steps with the time step being $2.5x10^{-6}$, and required 2087 CPU seconds on the Cray XMP. The residual reached a maximum of $3.4x10^{-1}$ and was reduced to a value of $5.1x10^{-3}$. The computation started with freestream conditions and a uniformly spaced (41x41) mesh. Thus the mesh

satisfied the mesh equation and was consistent with the solution. The tangential velocity boundary conditions at the plate were transitioned from the freestream value to an adherence value of zero using a cubic spline curve. The transition time was $3.351x10^{-3}$ seconds, which was the residence time. The mesh weighting function used in the mesh adaption was the gradient in (ξ, η) space of the Mach number. The adaption parameters were $\lambda_S = 1.0$, $\lambda_O = 10.0$, $\lambda_A = 10.0$, $\lambda_0 = 1.0$, $\lambda_1 = 5000.0$, and 30 smoothing passes were used. The mesh equations were iterated once for the predictor stage and once for the corrector stage. The mesh adapted quickly to the developing boundary layer. There was very little change in the mesh over the last two-thirds of the computation. Therefore, for steady flows, it would be more efficient if the mesh adaption was totally turned off after the flow was mostly developed. The adaption placed the first point from the plate at a distance of about $5.35x10^{-4}$.

Figure 10.28 shows the final mesh from the computation on the dynamically adaptive mesh. Figure 10.29 shows the u velocity profile at the location $x = 1.0$ meters. The comparison to the results from a compressible boundary layer code is excellent. Figure 10.30 shows the temperature profile. The comparison to the results from a compressible boundary layer code is excellent. The comparison of the v velocity profile is good; however, the freestream value of v from the computations does not match the value from the boundary layer code because the boundary layer code does not account for the leading edge shock.

The time history of the convergence showed a rather noisy covergence for the dynamically adaptive mesh computations. This is most likely due to the mesh dynamics. Near the end of the computations the convergence history bottomed out and exhibited significant noise. However, this did not threaten the stability of the

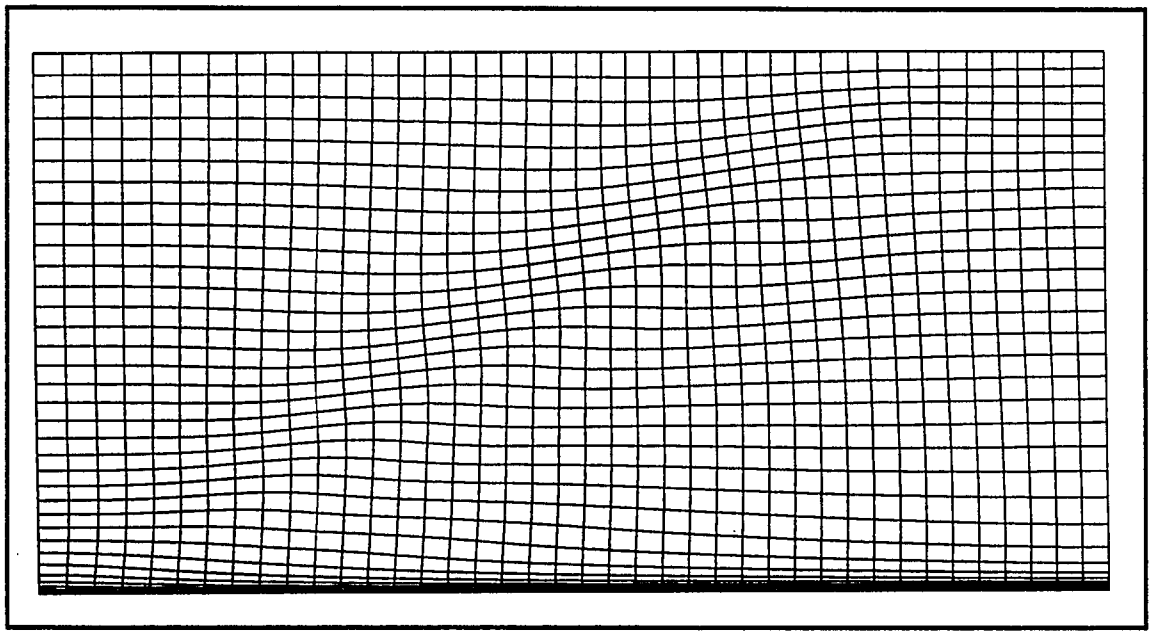

Figure 10.28: The final (41x41) dynamically adapted mesh for the supersonic boundary-layer computation using the time-differenced mesh-speeds

computation.

The values of the adaption parameters represent the highest values for which a reasonable computation could be performed. Higher values of λ_A were tried, but more smoothing passes were needed. It was felt that 30 smoothing passes was probably the most that should be done.

Computations on a dynamically adaptive mesh using the mesh speed equations were performed; however, they are not complete due to time limitations on this work and are not presented here.

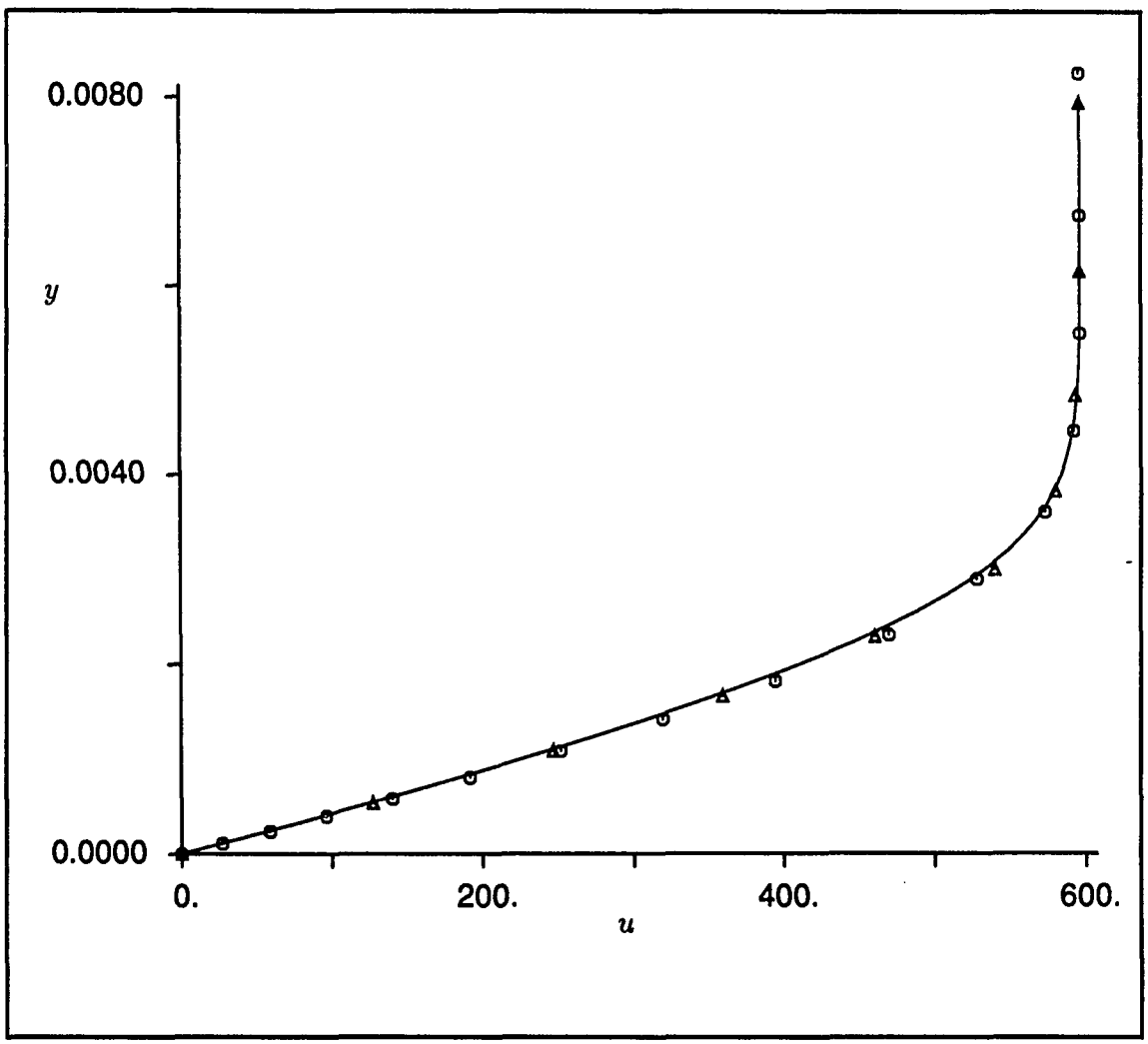

Figure 10.29: The u velocity profile at $x = 1.0$ units for the supersonic boundary-layer computation: (solid) boundary-layer code; (circles) static mesh; (triangles) dynamically adapted mesh

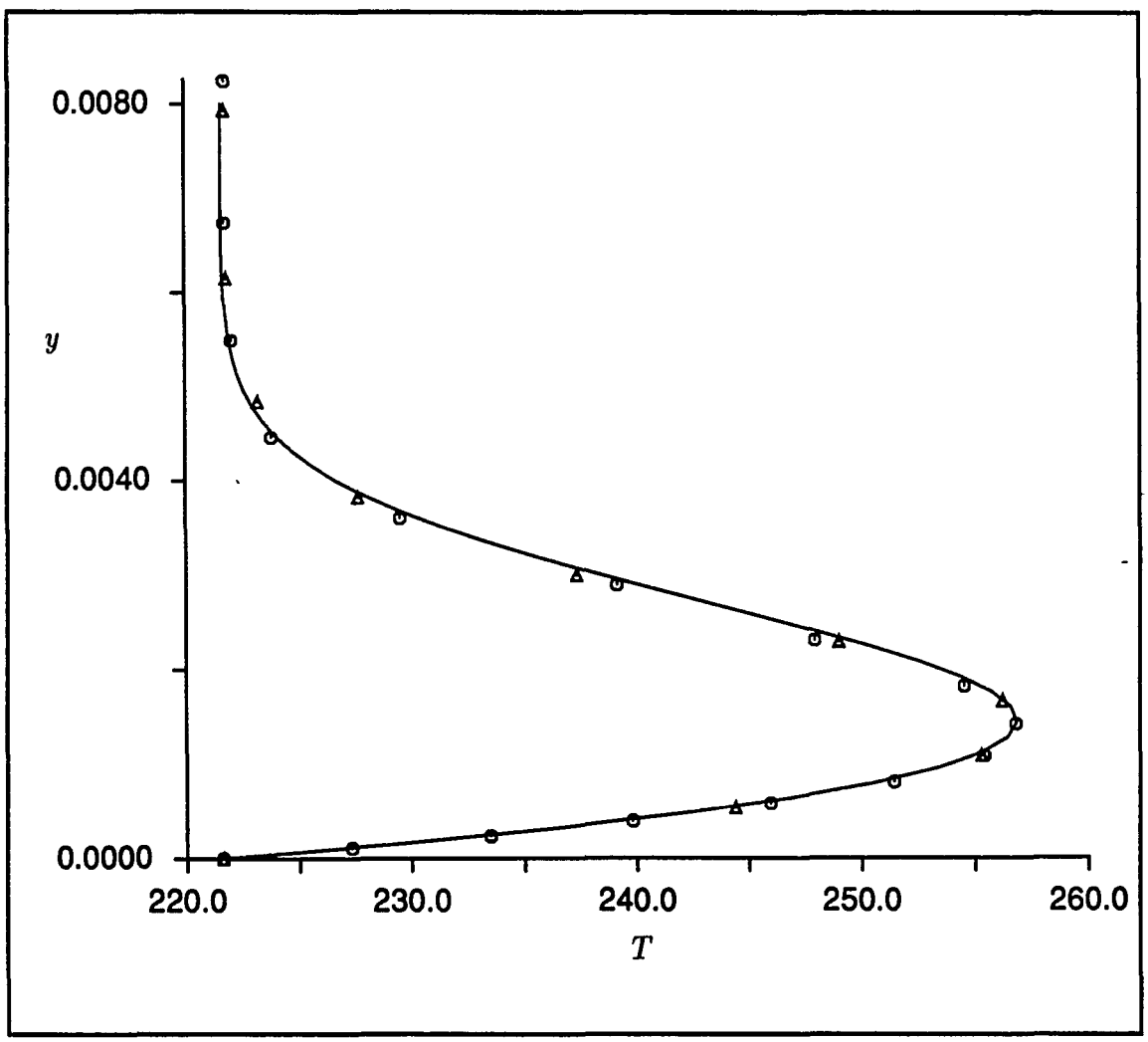

Figure 10.30: The temperature profile at $x = 1.0$ units for the supersonic boundary-layer computation: (solid) boundary-layer code; (circles) static mesh; (triangles) dynamically adapted mesh

The Shock/Boundary-Layer Interaction on a Flat Plate

This application of the dynamically adaptive mesh method examines the ability to adapt the mesh to flows involving a shock and a boundary layer in which part of the viscous flow is separated. An oblique shock impinges on a laminar boundary layer along a flat plate and pressure-gradient induced separation occurs. This problem was investigated experimentally by Hakkinen et al. [21]. The freestream conditions prior to the shock are a Mach number of 2.0, a Reynolds number of $2.96x10^5$ based on a length of 1.0 meters, and the flow is parallel to the plate. For the computations, a freestream temperature was set at 250 K.

The computation on the static mesh used a (75x65) mesh defined by Liou [36] in which the mesh spacings are based on triple-deck theory. The computational domain extends from $x = -0.23$ to $x = 2.0$ meters and from $y = 0$ to $y = 1.34196$ meters. The solution was initialized with the inviscid, supersonic flow which included the incident shock and reflected shock. The properties through the shocks were computed using oblique shock theory. The incident shock entered the flow domain through the supersonic inflow boundary, impinged on the plate at $x = 1.0$ meters, and the reflected shock left the computational domain through the supersonic outflow boundary. On the outflow boundary, in the subsonic portion of the boundary layer, subsonic outflow boundary conditions were applied with the pressure being obtained from the inviscid flow. The computation on the static mesh was run for 132000 time steps with a CFL number of 0.7, and required 20132 CPU seconds on the Cray XMP. The solution did not change significantly after about the 50000th time step and could have been stopped at that point.

The computation on the dynamically adaptive mesh using the time-differenced

mesh speeds required an initial mesh which satisfied the mesh equations and was consistent with the intial solution. The initial mesh was obtained by solving the mesh equations for the given adaption parameters for the problem. These adaption parameters were $\lambda_S = 1.0$, $\lambda_O = 1.0$, $\lambda_A = 1.0$, $\lambda_0 = 1.0$, $\lambda_1 = 500.0$, and 30 smoothing passes were used. The mesh adapted to the gradient of the Mach number in the (ξ, η) space. The velocity at the plate was transitioned from the inviscid flow velocity to zero velocity over a time interval of $3.95x10^{-3}$ seconds. The computation was performed for 21000 time steps with a CFL number of 0.6, and required 4143 CPU seconds on the Cray XMP.

Figure 10.31 shows the static and dynamically adapted meshes. Figure 10.32 shows the pressure contours for the computations. Figure 10.33 shows the comparison of the pressure coefficients along the plate. Figure 10.34 shows the comparison of the skin friction coefficient along the plate. Figure 10.35 shows the comparison of the u velocity profile in the boundary layer. The mesh spacing of the first mesh point off the wall for the static mesh was $1.5625x10^{-4}$ units while the mesh spacing for the dynamically adapted mesh was $2.8646x10^{-3}$ units. This difference in mesh resolution normal to the plate may explain the differences in the pressure and skin-friction coefficients.

Attempts to obtain more clustering normal to the wall for the dynamically adaptive mesh method relied on increasing the values of λ_A and λ_1 and were unsuccessful. This behavior is not yet understood. Perhaps using a different form of W would improve the resolution. Attempts at performing computations involving the use of the mesh speed equations were not successful and work had to be stopped due to time limitations.

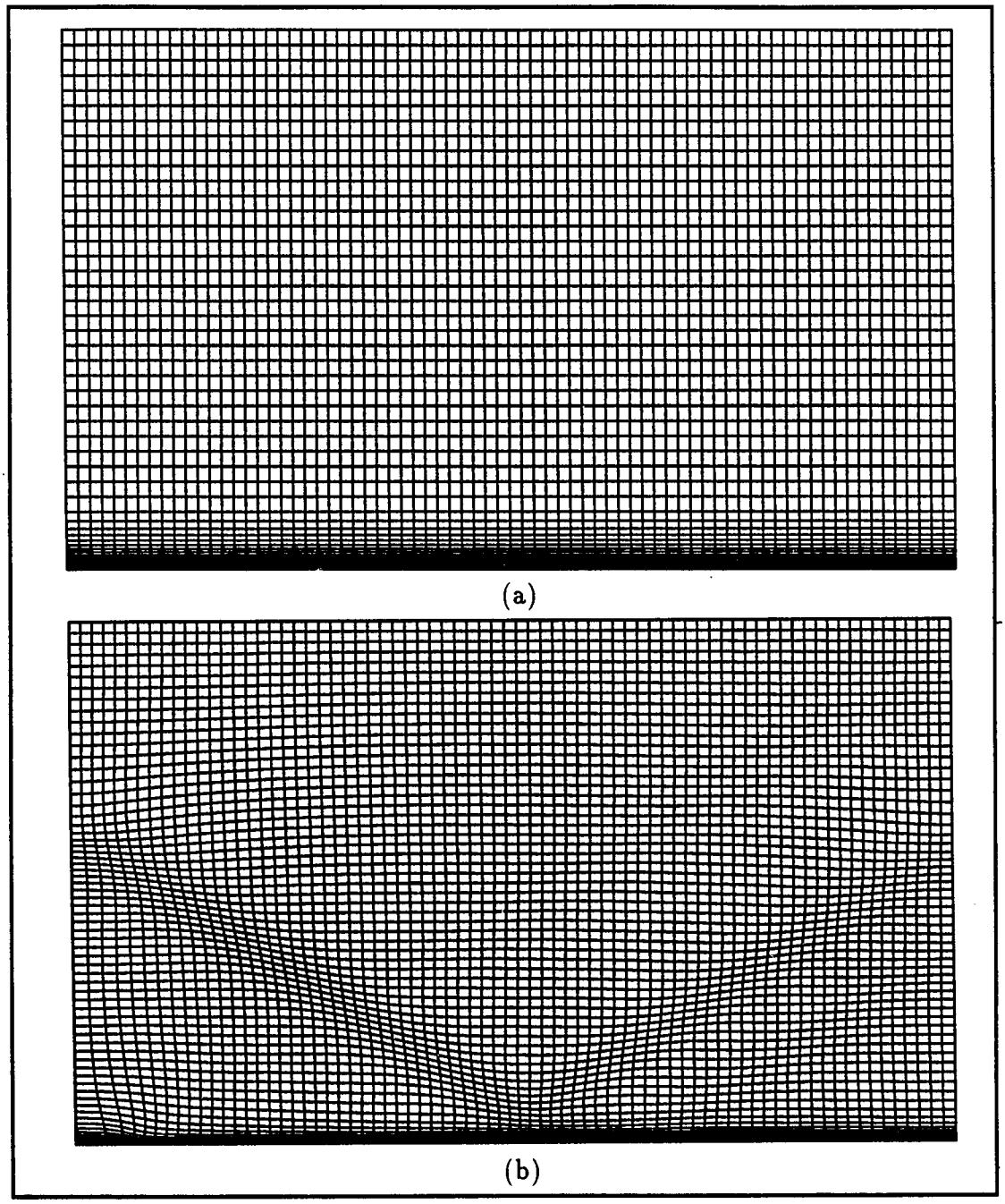

Figure 10.31: The (a) static and (b) dynamically adapted meshes for the shock/boundary-layer interaction problem

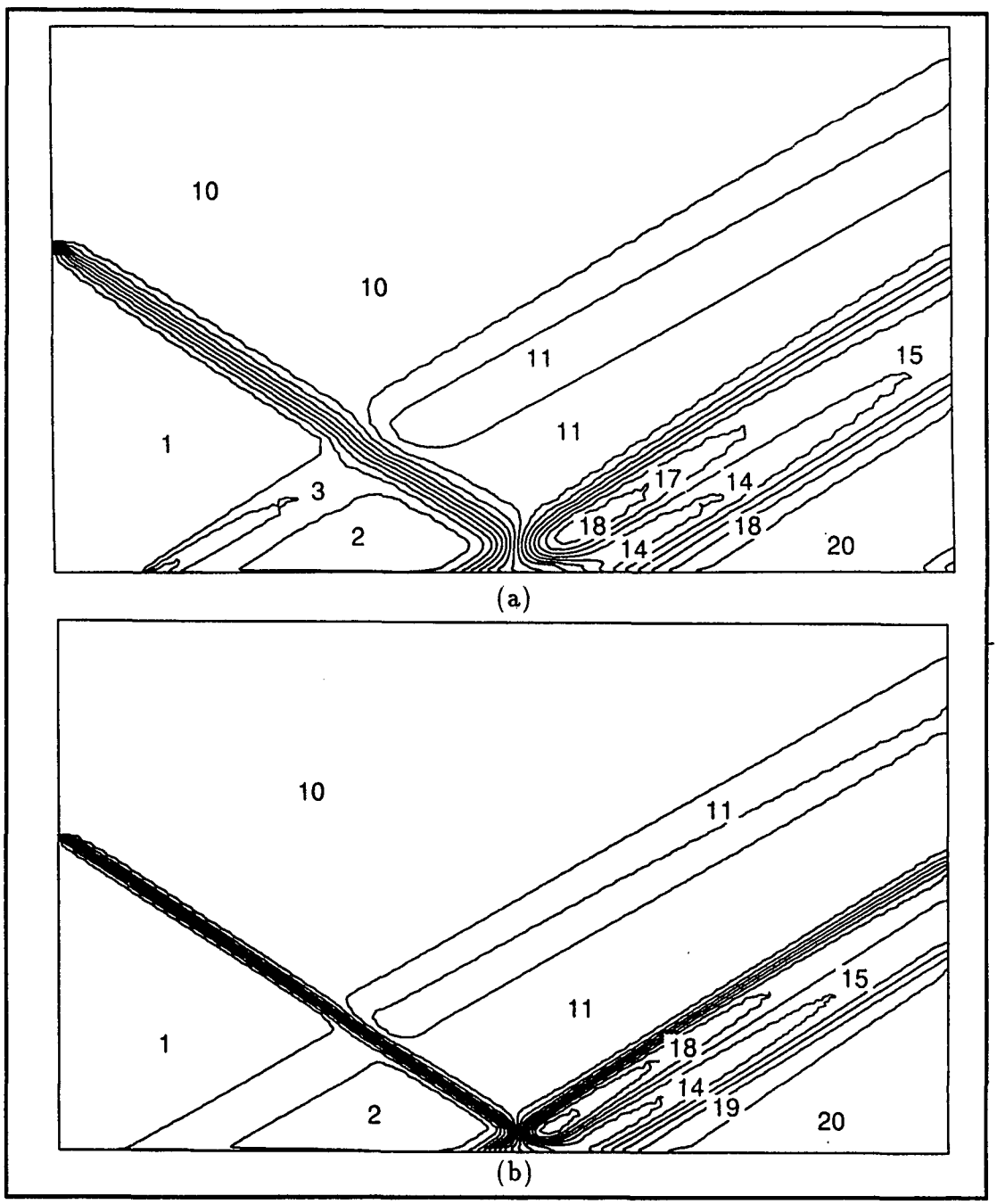

Figure 10.32: The pressure contours for the shock/boundary-layer interaction problem: (a) static mesh; (b) dynamically adapted mesh (higher number indicates higher pressure)

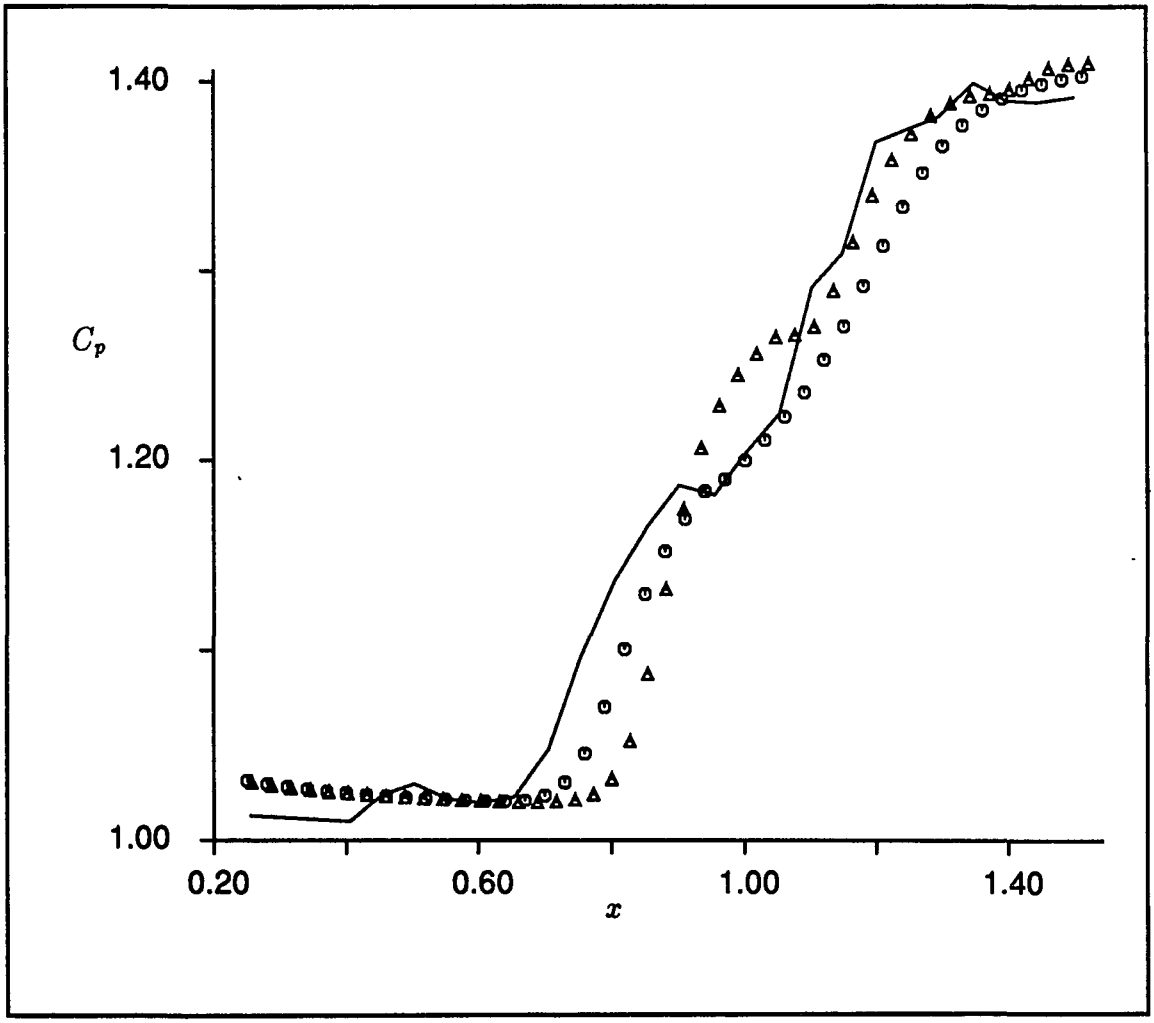

Figure 10.33: The pressure coefficients along the plate for the shock/boundary-layer interaction problem: (solid) experimental data; (circles) static mesh; (triangles) dynamically adapted mesh

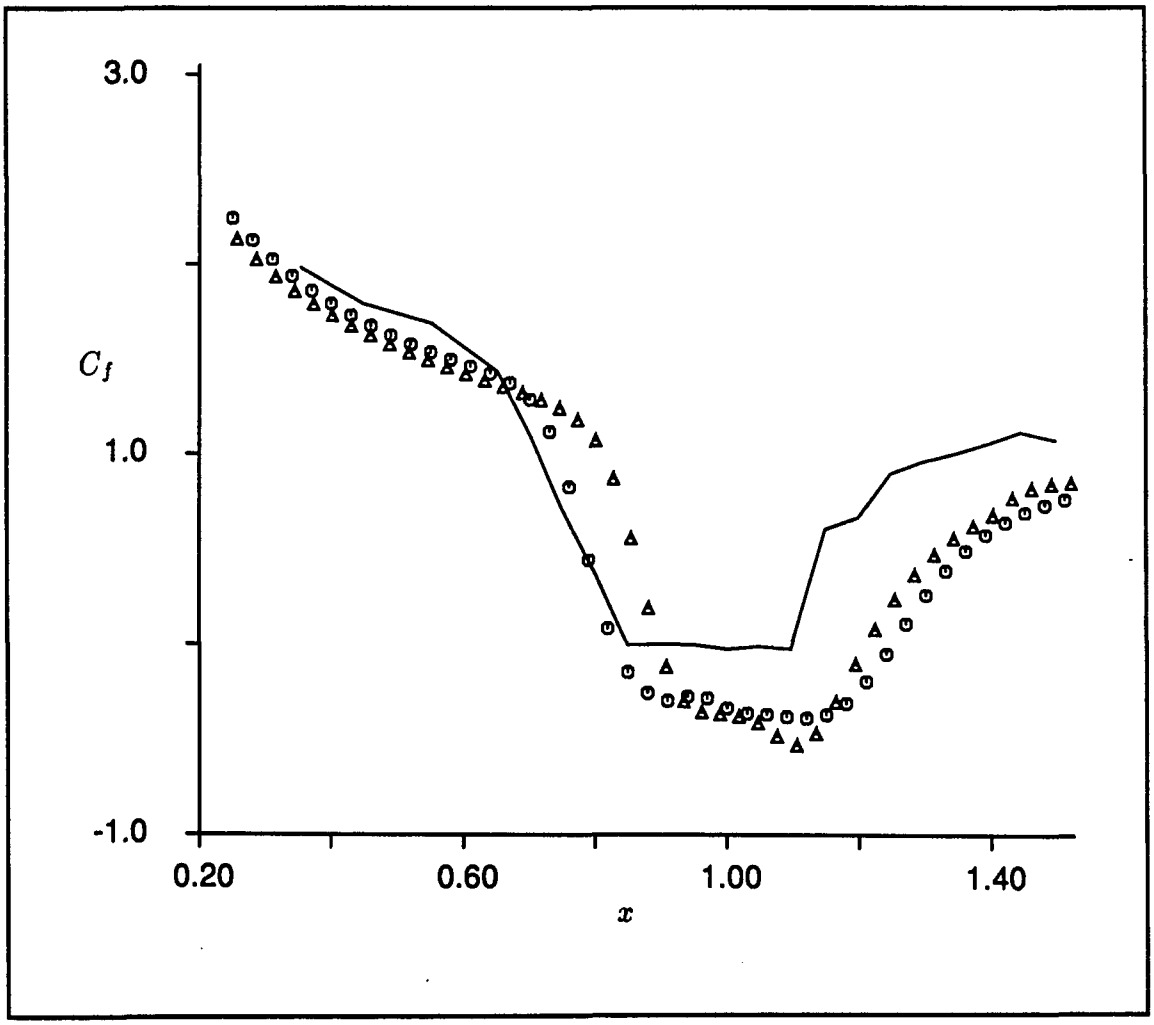

Figure 10.34: The skin friction coefficients along the plate for the shock/boundary-layer interaction problem: (solid) experimental data; (circles) static mesh; (triangles) dynamically adapted mesh

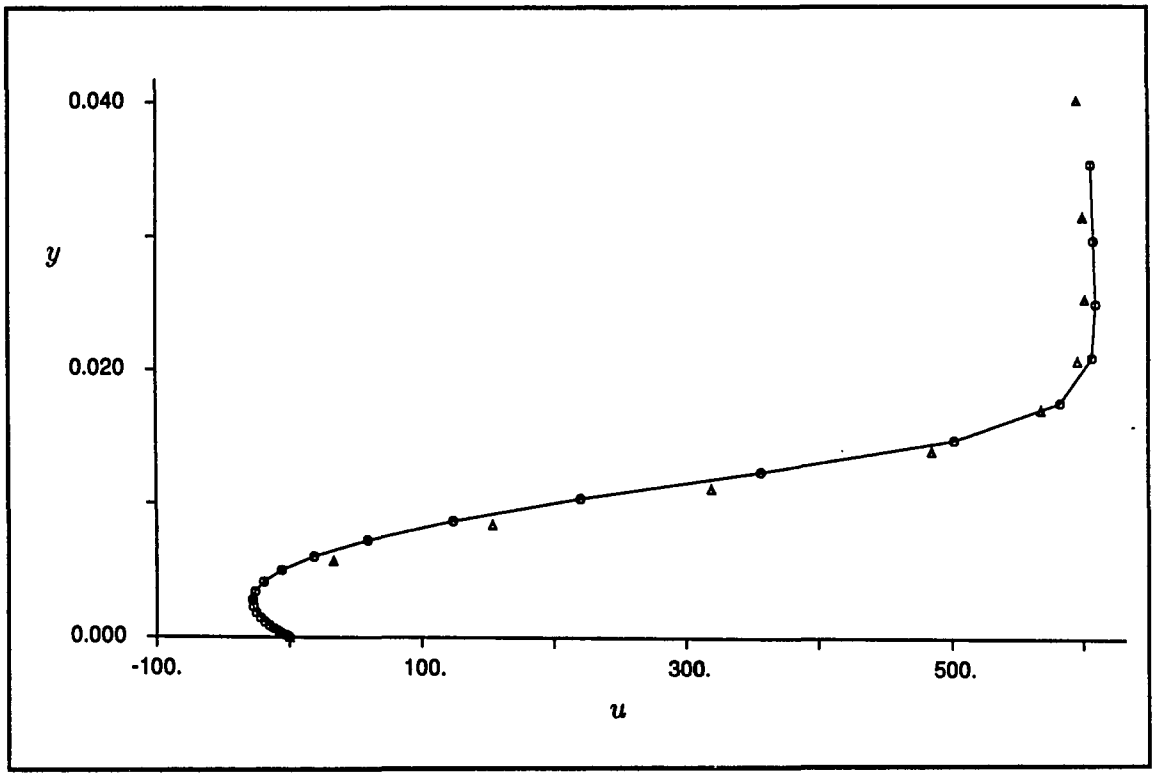

Figure 10.35: The u velocity profile at the location $x = 1.0$ for the shock/boundary-layer interaction problem: (solid and circles) static mesh; (triangles) dynamically adapted mesh

The Transonic Diffuser Flow

A complex internal flow combining the flow features discussed previously is the viscous flow through a transonic diffuser. For a range of exit pressures, there exists a shock in the diverging portion of the inlet. The exit pressure may be such that a strong shock exists which causes the separation of the flow from the inlet walls. This flow has been studied extensively by Bogar et al. [11].

For a ratio of exit pressure to inlet total pressure of $R_p = 0.82$, the flow in the inlet is steady. A shock forms in the diverging portion of the inlet; however, it is not strong enough to cause flow separation. The inflow conditions included a total pressure of $p_t = 135000\ N/m^2$ and a total temperature of $T_t = 292.0\ K$. The walls were adiabatic. The Baldwin-Lomax turbulence model was also used.

An (81x61) mesh was computed for the static mesh computations. The mesh equations were solved with the mesh parameters of $\lambda_S = 1.0$, $\lambda_0 = 1.0$, $\lambda_A = 0.0$. The constant-η mesh lines were then clustered near the lower and upper walls of the inlet by using the Roberts stretching function with a stretching parameter of 1.0005. The stretching was performed along constant-ξ mesh lines. The static mesh is shown in figure 10.36. The inflow conditions were used to specify a uniform initial solution on the static mesh. A CFL number of 0.6 was used for a computation of an estimated 10 CPU hours on the Cray XMP. The final residual was a value of $2.19x10^{-2}$. Figure 10.37 shows the Mach number contours for the diffuser. Figures 10.39 and 10.40 show the static pressures along the bottom and top walls, respectively, as compared to the experimental data. The static pressures are normalized by the inflow total pressure. The comparisons are very good.

A computation was also performed with a dynamically adaptive mesh using

time-differenced mesh speeds. An initial mesh was generated with the parameters $\lambda_S = 1.0$, $\lambda_O = 1.0$, $\lambda_A = 1.0$, $\omega_G = 1.8$, and $\omega_{GB} = 0.9$. The initial solution was specified as the total pressure and temperature with u and v equal to zero. The exit pressure was then decreased using a cubic spline curve with zero end-slope conditions to result in a final exit pressure of $p_{exit} = 110700\ N/m^2$ or $R_p = 0.82$. As the exit pressure was reduced, the flow was accelerated and the boundary layers developed. The adaption of the mesh was driven by the gradient of the Mach number in the (ξ, η) space. The mesh adaption parameters were $\lambda_1 = 500$ and 30 smoothing passes were made for each computation of W. Figure 10.38 shows the Mach number contours at the end of the computation. It is estimated that 10 CPU hours on the Cray XMP were required to obtain these results. The exact computational cost is difficult to determine because as errors in the code were discovered, the computation proceded from the latest computation. Figures 10.39 and 10.40 show the static pressures along the bottom and top walls, respectively, as compared to the experimental data and the static mesh results. The adapted mesh was unable to achieve the mesh spacing at the wall of the static mesh and this resulted in significant errors in computing the boundary layers and this caused no shock to form. The reason for the poor resolution of the boundary layers is not understood. The time limitations of this work prevented further study.

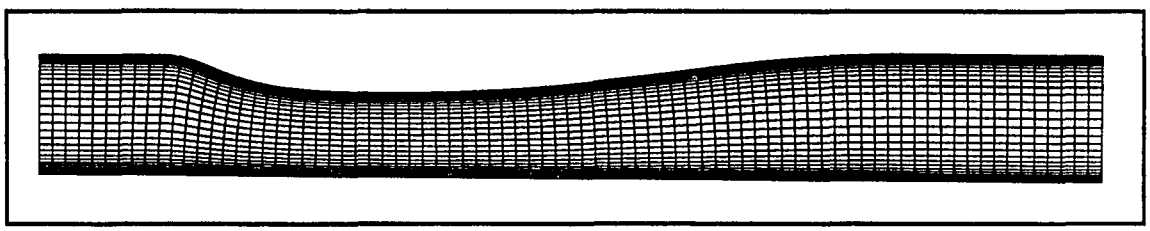

Figure 10.36: The (81x61) mesh used for the static mesh calculations of the turbulent flow in the transonic diffuser

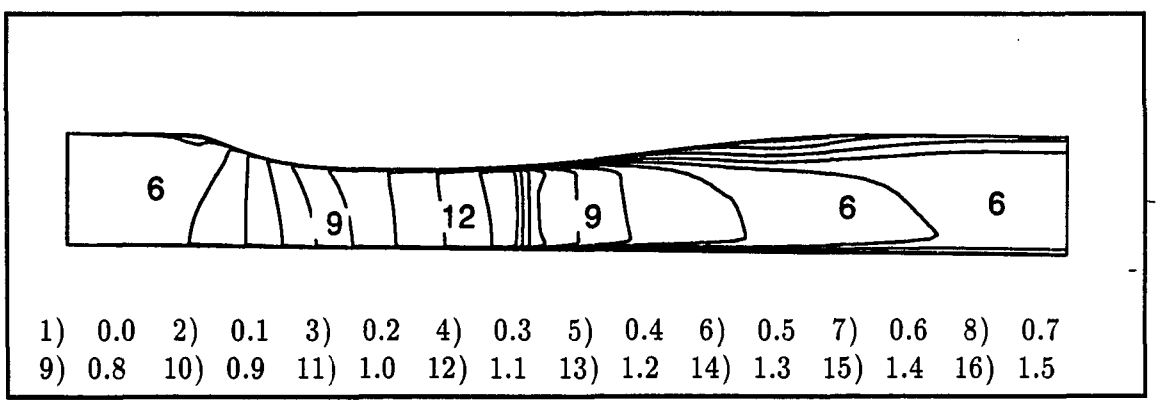

Figure 10.37: The Mach number contours for the static mesh computations of the turbulent flow in the transonic diffuser

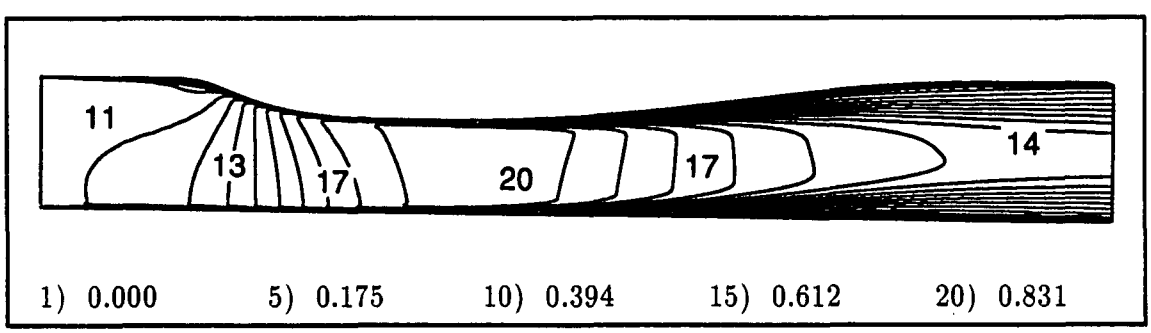

Figure 10.38: The Mach number contours for the dynamically adapted mesh computations of the turbulent flow in the transonic diffuser

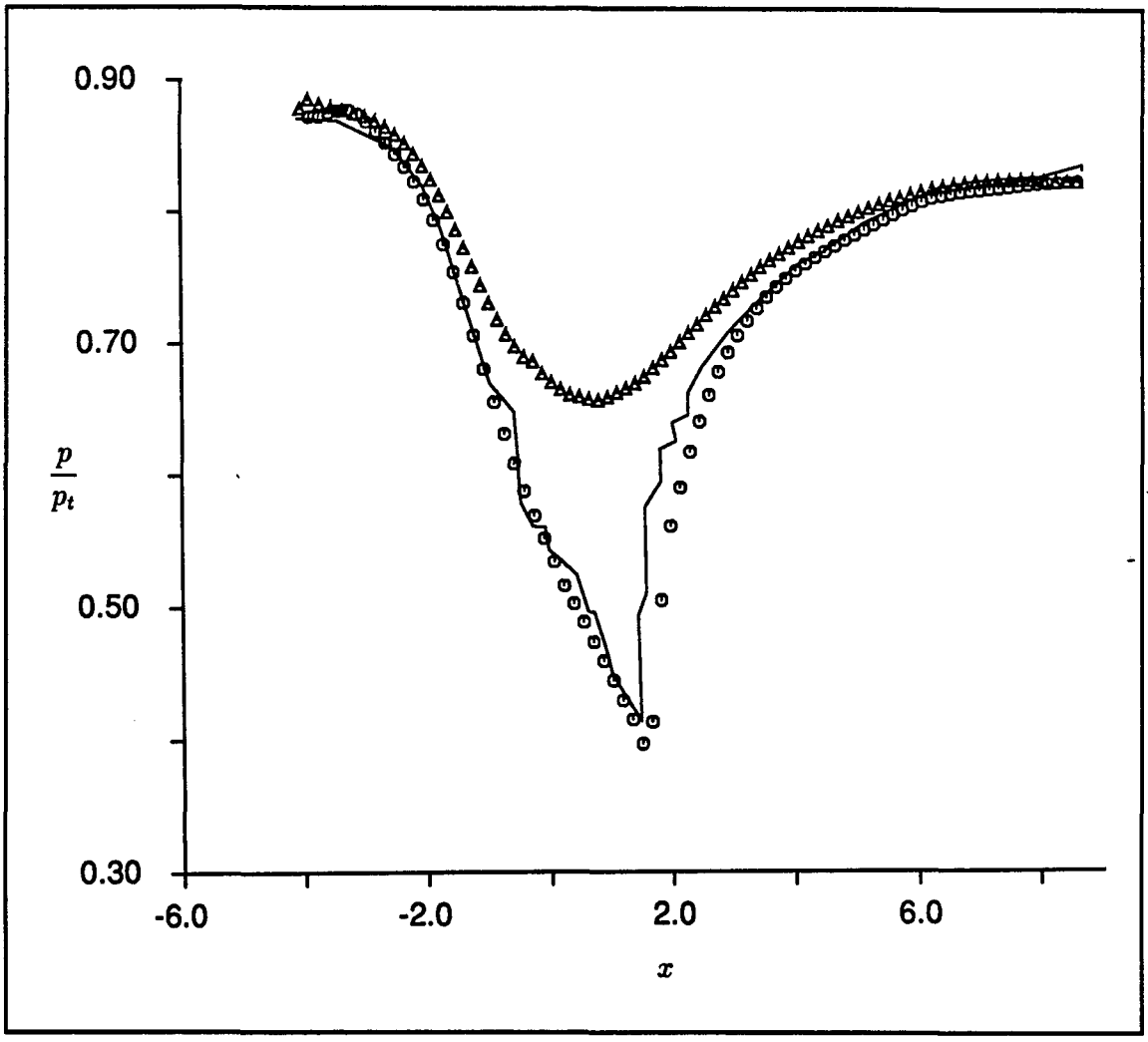

Figure 10.39: The pressure ratios along the bottom wall for the turbulent flow in the transonic diffuser: (solid) experimental data; (circles) static mesh; (triangles) dynamically adapted mesh

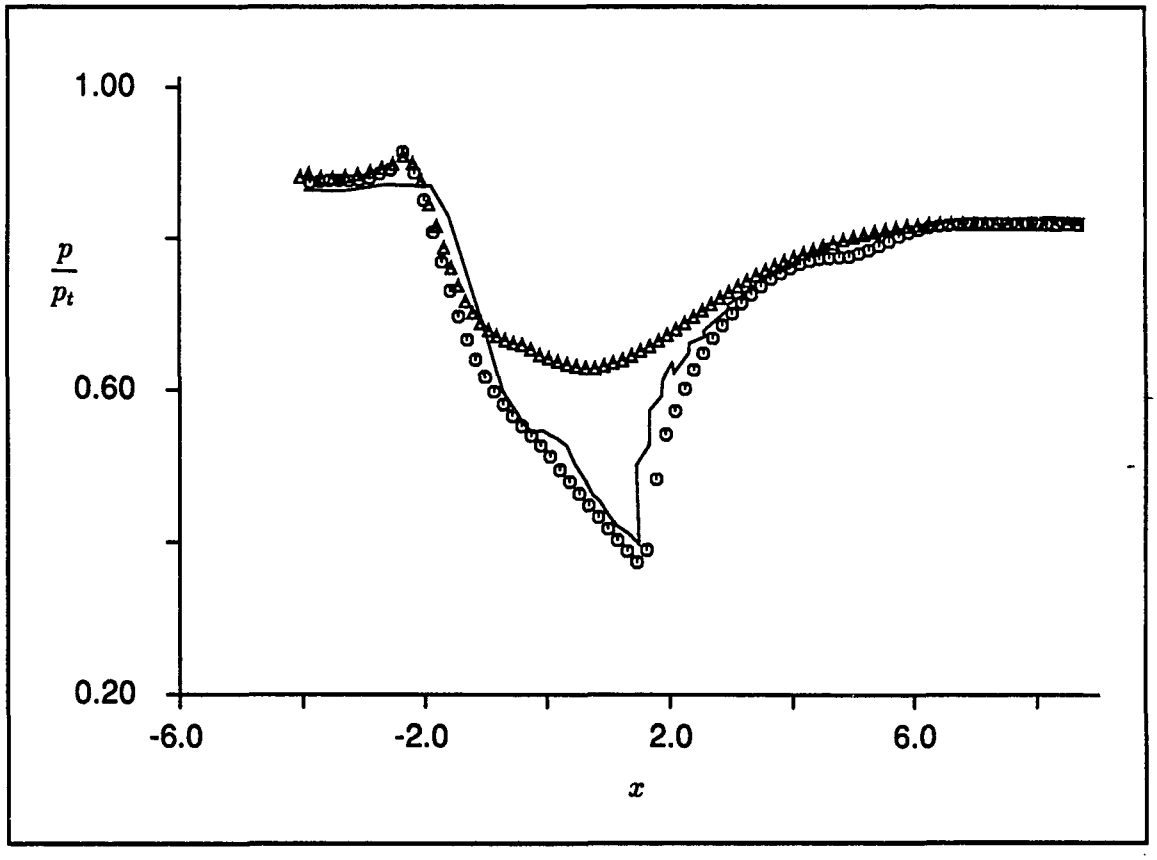

Figure 10.40: The pressure ratios along the upper wall for the turbulent flow in the transonic diffuser: (solid) experimental data; (circles) static mesh; (triangles) dynamically adapted mesh

CHAPTER 11. SUMMARY AND CONCLUSIONS

The material presented here has examined dynamically adaptive meshes involving mesh and mesh speed equations developed from variational principles. The approach of Brackbill and Saltzman [12] has been modified slightly to account for the scaling of the integral equations. The mesh speed equations were then derived based on a time differentiation of the mesh equations. The results from numerical experiments involving model problems showed the behavior of the mesh and mesh speed equations.

The mesh equations provided the desired qualities; however, the meshes involved compromises between the qualities of smoothness, orthogonality, and volume adaption. It does appear that there is a limit to the values of λ_O and λ_A as one attempts to obtain an orthogonal or adaptive mesh.

The solutions of the mesh speed equations do show a potential application for computing mesh speeds for the case of boundary motion. The computed mesh speeds are able to produce meshes which maintain the desired smoothness, orthogonality, and volume adaption qualities. Also, the orthogonality of the mesh lines at the boundaries is maintained. One future effort could be in applying the mesh speed equations for problems involving dynamic boundaries.

The solutions of the mesh speed equations for the model problems also demon-

strated some capability to dynamically adapt the mesh to time-dependent solutions. However, the level of adaption for discontinuities was not as great as one would hope. Further, there seemed to be difficulty in computing mesh speeds as the level of adaption increased.

The mesh control law was demonstrated to be effective in damping errors in the mesh speeds. There appears to be a limit on the value of λ_C for which a solution to the mesh speed equations can be obtained.

Dynamic solution adaption was demonstrated for unsteady flowfields. The results showed that computing the mesh speeds from the mesh speed equations was more accurate than computing the mesh speeds from a backwards time difference of the mesh. The resolution of the shocks in the converging-diverging nozzle was improved. There appeared to be a limit on the mesh clustering that can be achieved for the resolution of boundary layers. For some cases, it was not possible to obtain the proper clusterings for the boundary layers and this resulted in significant errors.

Using dynamically adaptive meshes was significantly more computationally expensive than using a static mesh. The problems presented in the present work attempted mainly to demonstrate the validity of the approach. Static meshes are probably adequate for all of the flowfields. The approach would probably be more applicable for problems in which the solution changes significantly in space and in time.

BIBLIOGRAPHY

[1] Anderson, D.A. "Adaptive Grid Methods for Partial Differential Equations." In Advances in Grid Generation. Editors K. Ghia and U. Ghia. Presented at the Applied Mechanics, Bioengineering, and Fluids Engineering Conference, June 20-22, Houston, Texas, 1983. New York: ASME, 1983. 1-15.

[2] Anderson, D.A. "Adaptive Mesh Schemes Based on Grid Speeds." AIAA Paper 83-1931. Presented at the Sixth AIAA Computational Fluid Dynamics Conference.

[3] Anderson, D.A. "Application of Adaptive Grids to Transient Problems." In Adaptive Computational Methods for Partial Differential Equations. Editors I. Babuska, J. Chandra, and J. Flaherty. Philadelphia: SIAM, 1983. 208-223.

[4] Anderson, D.A. and M.M. Rai. "The Use of Solution Adaptive Grids in Solving Partial Differential Equations." Numerical Grid Generation. Ed. J.F. Thompson. New York: North-Holland, 1982. 317-338.

[5] Anderson, D.A., J.C. Tannehill, and R.H. Pletcher. Computational Fluid Mechanics and Heat Transfer. New York: McGraw-Hill Book Company, 1984.

[6] Baines, M.J. "Moving Element Methods for Time Dependent Problems." Numerical Methods for Fluid Dynamics III. Ed. K.W. Morton and M.J. Baines. Clarendon Press, Oxford, 1988. 513-518.

[7] Baldwin, B.S. and H. Lomax. "Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows." AIAA Paper 78-0257. Presented at the 16th AIAA Aerospace Sciences Meeting, Huntsville, Alabama, January 16-18, 1978.

[8] Beck, B.C. "A Dynamically Adaptive Grid Scheme Applied to a 1-D Shock Tube." M.S. Thesis, Iowa State University, Ames, Iowa, 1992.

[9] Benson, R. and D. McRae. "A Three-Dimensional Dynamic Solution-Adaptive Mesh Algorithm." AIAA Paper 90-1566. Presented at the AIAA 21st Fluid Dynamics, Plasma Dynamics and Lasers Conference, Seattle, Washington, June 18-20, 1990.

[10] Bockelie, M.J., P.R. Eiseman, and R.E. Smith. "An Adaptive Grid Method for Computing Time Accurate Solutions on Structured Grids." AIAA Paper 91-0144. Presented at the AIAA 29th Aerospace Sciences Meeting, Reno, Nevada, January 7-10, 1991.

[11] Bogar, T.J., M. Sajben, and J.C. Kroutil. "Characteristic Frequency and Length Scales in Transonic Diffuser Flow Oscillations." AIAA Journal 21 (1983): 1232-1240.

[12] Brackbill, J.U. and J.S. Saltzman. "Adaptive Zoning for Singular Problems in Two Dimensions." Journal of Computational Fluid Dynamics 46 (1982): 342-368.

[13] Carcaillet, R. "Optimization of Three-Dimensional Computational Grids and Generation of Flow Adaptive Computational Grids." AIAA Paper 86-0156. Presented at the AIAA 24th Aerospace Sciences Meeting, Reno, Nevada, January 6-9, 1986.

[14] Castillo, J.E., ed. Mathematical Aspects of Numerical Grid Generation. Philadelphia: SIAM, 1991.

[15] Connett, W.C., R.K. Agarwal, and A.L. Schwartz. "An Adaptive Grid-Generation Scheme for Flowfield Calculation." AIAA Paper 87-0199. Presented at the AIAA 25th Aerospace Sciences Meeting, Reno, Nevada, January 12-15, 1987.

[16] Deiwert, G.S., E. Venkatapathy, C. Davies, J. Djomehri, and K. Abrahamson. "Application of a Self-Adaptive Grid Method to Complex Flows." NASA TM-102223, July, 1989.

[17] Djomehri, M.J. and G.S. Deiwert. "Three-Dimensional Self-Adaptive Grid Method for Complex Flows." NASA TM-101027, November 1988.

[18] Dorfi, E.A. and L. O'C. Drury. "Simple Adaptive Grids for 1-D Initial Value Problems." Journal of Computational Physics 69 (1987): 175-195.

[19] Eiseman, P.R. and G. Erlebacher. "Grid Generation for the Solution of Partial Differential Equations." ICASE Report No. 87-57. NASA Langley Research Center, August 1987.

[20] Greenberg, J.B. "A New Self-Adaptive Grid Method." AIAA Journal 23 (1985): 317-320.

[21] Hakkinen, R.J., I Greber, L. Trilling, and S.S. Abarbanel. "The Interaction of an Oblique Shock Wave with a Laminar Boundary Layer." NASA Memo 2-18-59W, 1959.

[22] Harten, A. and J.M. Hyman. "Self-Adjusting Grid Methods for One-Dimensional Hyperbolic Conservation Laws." Journal of Computational Fluid Dynamics 50 (1983): 235-269.

[23] Hawken, D.F., J.J. Gottlieb, and J.S. Hansen. "Review of Some Adaptive Node-Movement Techniques in Finite-Element and Finite-Difference Solutions of Partial Differential Equations." Journal of Computational Physics 95 (1991): 254-302.

[24] Hindman, R.G. "Generalized Coordinate Forms of Governing Fluid Equations and Associated Geometrically Induced Errors." AIAA Journal 20 (1982): 1359-1367.

[25] Hindman, R.G. "Notes from Aer E 690K, Static and Dynamic Grid Generation." Iowa State University, Spring 1990.

[26] Hindman, R.G. "On Shock Capturing Methods and Why They Work." AIAA Paper 88-0622. Presented at the AIAA 26th Aerospace Sciences Meeting, Reno, Nevada, January 11-16, 1988.

[27] Hindman, R.G. Personal communication, July 1990.

[28] Hindman, R.G., P. Kutler, and D.A. Anderson. "Two-Dimensional Unsteady Euler-Equation Solver for Arbitrarily Shaped Flow Regions." AIAA Journal 19 (1981): 424-431.

[29] Hindman, R.G. and J. Spencer. "A New Approach to Truly Adaptive Grid Generation." AIAA Paper 83-0450. Presented at the AIAA 21st Aerospace Sciences Meeting, Reno, Nevada, January 10-13, 1983.

[30] Hirsch, C. Numerical Computation of Internal and External Flows, Volume 2: Computational Methods for Inviscid and Viscous Flows. New York: John Wiley & Sons, 1990.

[31] Holcomb, J.E. "Development of an Adaptive Grid Navier-Stokes Analysis Method for Rocket Base Flows." AIAA Paper 88-2905. Presented at the AIAA/ASME/SAE/ASEE 24th Joint Propulsion Conference, Boston, Massachusetts, July 11-13, 1988.

[32] Holcomb, J.E. and R.G. Hindman. "Development of a Dynamically Adaptive Grid Method for Multidimensional Problems." AIAA Paper 84-1668. Presented at the AIAA 17th Fluid Dynamics, Plasma Dynamics, and Lasers Conference, Snowmass, Colorado, June 25-27, 1984.

[33] Kreis, R.I., F.C. Thames, and H.A. Hassan. "Application of a Variational Method for Generating Adaptive Grids." AIAA Journal 24 (1986): 404-410.

[34] Lax, P.D. "Weak Solutions of Nonlinear Hyperbolic Equations and their Numerical Computation." Communications on Pure and Applied Mathematics 7 (1954): 159-93.

[35] Liou, M.-S. "A Generalized Procedure for Constructing an Upwind-Based TVD Scheme." AIAA Paper 87-0355. Presented at the AIAA 25th Aerospace Sciences Meeting, Reno, Nevada, January 12-15, 1987.

[36] Liou, M-S. "A Newton/Upwind Method and Numerical Study of Shock Wave/Boundary Layer Interactions." International Journal for Numerical Methods in Fluids 9 (1989): 747-761.

[37] Liou, M.-S. and A.T. Hsu. "A Time-Accurate Finite Volume High Resolution Scheme for Three Dimensional Navier-Stokes Equations." AIAA Paper 89-1994. Presented at the AIAA 9th Computational Fluid Dynamics Conference, Buffalo, New York, June 13-15, 1989.

[38] Rai, M.M. "A Philosophy for Construction of Solution Adaptive Grids." Ph.D. Dissertation, Iowa State University, Ames, Iowa, 1982.

[39] Roache, P.J. and S. Steinberg. "A New Approach to Grid Generation Using a Variational Formulation." AIAA Paper 85-1527. Presented at the Seventh AIAA Computational Fluid Dynamics Conference.

[40] Roe, P.L. "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes." Journal of Computational Physics 43 (1981): 357-372.

[41] Thomas, P.D. and C.K. Lombard. "Geometric Conservation Law and Its Application to Flow Computations on Moving Grids." AIAA Journal 17 (1979): 1030-1037.

[42] Thompson, J.F. "Some Current Trends in Numerical Grid Generation." Numerical Methods for Fluid Dynamics III. Ed. K.W. Morton and M.J. Baines. Oxford: Clarendon Press, 1988. 87-100.

[43] Thompson, J.F. "A Survey of Dynamically-Adaptive Grids in the Numerical Solution of Partial Differential Equations." AIAA Paper 84-1606. Presented at

the AIAA 17th Fluid Dynamics, Plasma Dynamics, and Lasers Conference, Snowmass, Colorado, June 25-27, 1984.

[44] Thompson, J.F., Z.U.A. Warsi, and C.W. Mastin. Numerical Grid Generation, Foundations and Applications. New York: North-Holland, Elsevier Science Publishing, Inc., 1985.

[45] Vinokur, M. "An Analysis of Finite-Difference and Finite-Volume Formulations of Conservation Laws." Journal of Computational Physics. 81 (1989): 1-52.